DotNet Campania | team

Windows Workflow Foundation

28 Gennaio 2012

Felice Pescatore – felice.pescatore@gmail.com
www.felicepescatore.it - www.storiainformatica.it
felice.pescatore @felicepescatore Felice Pescatore

DotNet Campania | team

Agenda

Mattina (10.00 – 13.00)

- **I Workflow** (parte 1)
- **I Workflow nel mondo .NET** (parte 2)
  - Imperative Code vs. Declarative Design
  - Tipologie di Workflow
  - Activities
  - Activities Asincrone
  - Built-in Activities
    - Rehosting Workflow Designer
  - Esecuzione di un Workflow

Pomeriggio (14.00 -17.00)

- **Workflow Services** (parte 3)
- **Aspetti Avanzati** (parte 4)
  - Error Handling, Exception e Fault
  - Windows Server AppFabric
    - Monitoring
    - Persistenza
    - Cache
  - Security
    - Introduzione a WIF, WINDOWS IDENTITY FOUNDATION
    - WFF e WIF

Conclusioni

Chi sono e di cosa mi occupo

Personalmente amo definire la professione di Ingegnere come una professione "creativa" basata sulla ricerca della soluzione più innovativa e performante per il problema affrontato.

L'Ingegnere non deve restringere il dominio del problema per la ricerca della soluzione ma ampliarlo il più possibile perché nuove alternative possano rilevarsi e nuovi obiettivi porsi.

Ovviamente bisogna essere in grado di bilanciare quelle che sono le esigenze del mondo reale e quelle che è la naturale spinta verso la soluzione più raffinata: ecco dove l'esperienza, le competenze e la professionalità raggiunta fanno la differenza.

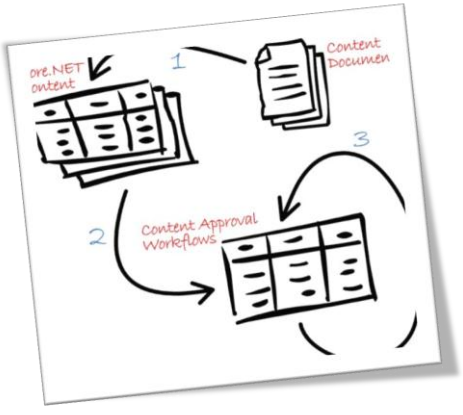
Nell'Ingegneria del Software tutto questo si arricchisce di un fattore aggiuntivo: spesso le metodologie applicate sono empiriche, cioè prodotte dai risultati sul campo, e difficilmente distillabili in formulari o regole precise.

Un ingegnere informatico utilizza l'informatica per risolvere problemi complessi, pianificando, ottimizzando le risorse e calandosi nel dominio applicativo.

Attualmente lavoro presso la *Microgame Spa* di Benevento, società specializzata nella realizzazione di piattaforme per il gambling (scommesse) online, dove mi occupo di architetture di integrazione con partner esteri.

Microsoft CERTIFIED Professional Developer | Enterprise Applications Developer

Parte Prima



I WORKFLOW

DotNet Campania team

Il mondo dei Workflow

Un workflow è un modo strutturato (visuale) per descrivere una serie di attività, il loro ordine di esecuzione e le relazioni che ci sono tra loro, in modo da modellare puntualmente un task/lavoro.

> "Workflow is the operational aspect of a work procedure: how tasks are structured, who performs them, what their relative order is, how they are synchronized, how information flows to support the tasks and how tasks are tracked."

http://en.wikipedia.org/wiki/Workflow

DotNet Campania team

Il mondo dei Workflow

- Si tratta di una serie di step (passi) orchestrati per realizzare un business process
- Può necessitare dell'intervento umano (attesa di verifica, risposta ad una notifica, ecc.)
- Può coordinare software/servizi terzi
- Tipicamente è descrivibile attraverso un **flowchart** o uno **State Diagram**
- Può essere short-running o long-running, anche se in quest'ultimo caso da il meglio di se.

Automate existing business processes

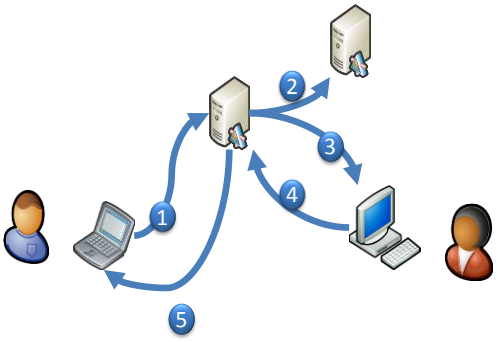
DotNet Campania team
Il mondo dei Workflow

- Coordinamento di attività (work) eseguite da software e/o persone
- Gestione ottimale di processi ***long running and stateful***
- Modelli estensibili (***extensible models)***
- Ciclo di vita ***trasparente e dincamico***

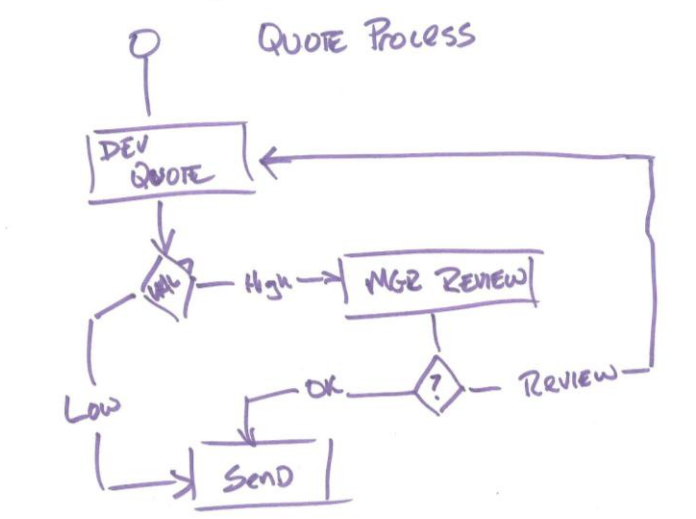
Windows Workflow Foundation **Parte Prima**

Long running logical processes that are episodic in nature.

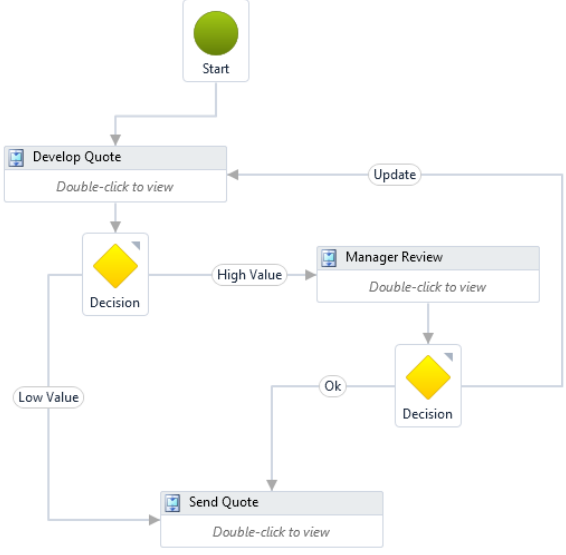
- Processi di approvazione
- Elaborazioni di grandi quantità di dati con breakpoint;
- Processi periodici in background:
  - Invio di email di alert
  - Sincronizzazione dati
  - Refresh della cache



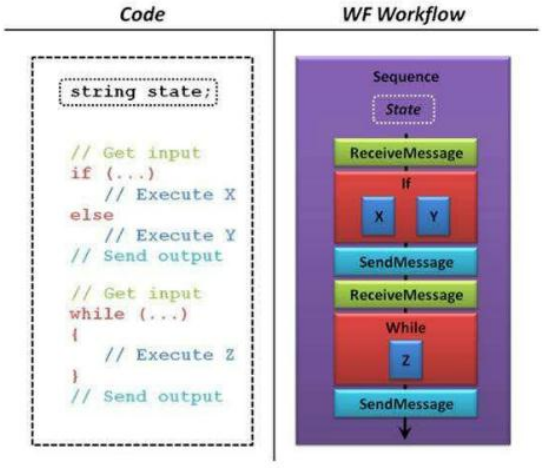
Il mondo dei Workflow: sulla carta….



… sul calcolatore



Focus su **COSA** (*What)* e non sul **COME** (*How)*



Microsoft .NET | Windows Workflow Foundation

I WORKFLOW NEL MONDO .NET

A workflow is a program

DotNet Campania team
Workflow nel mondo .NET

A workflow is a declarative program

COSA E NON COME

Windows Workflow Foundation

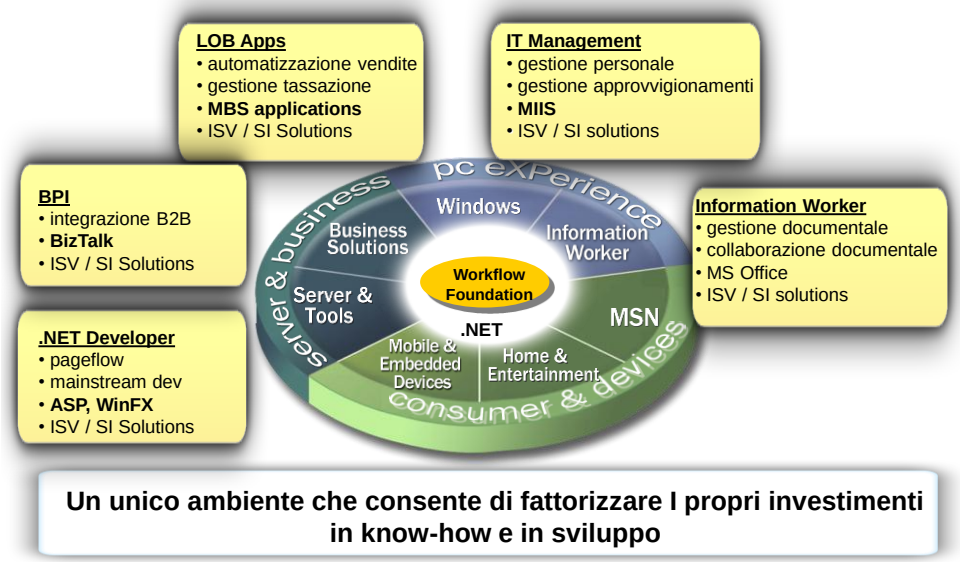
L'implementazione Microsoft dell'ecosistema Workflow nasce con il .Net Framework 3.0 e acquista il nome di Windows Workflow Foundation (WF o WFF)

Si tratta di un framework per lo sviluppo rapido (?) di applicazioni WF-based che ingloba:

- Un modello di programmazione estendibile basato su markup XAML
- Un potete engine corredato da una completa infrastruttura che ne permette l'utilizzo in scenari eterogenei.

Inoltre non potevano mancare i tool visuali di sviluppo integrati in VS 2005 e VS 2008.

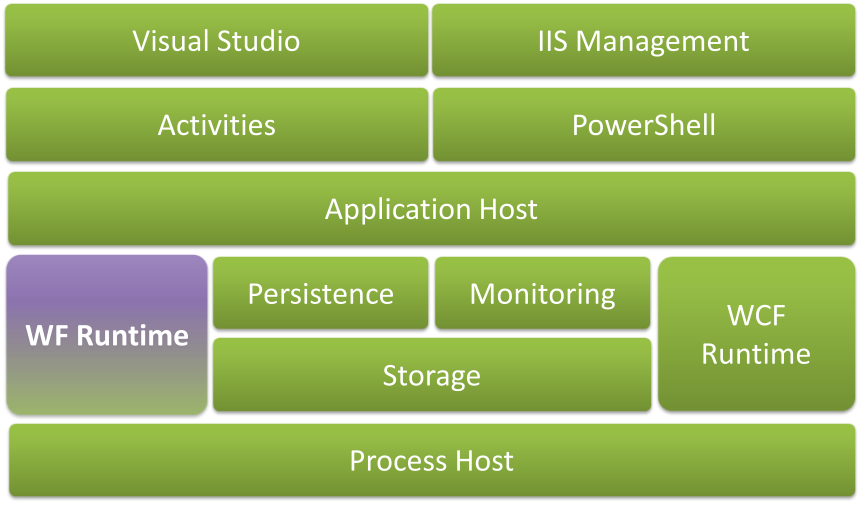
Windows Workflow Foundation



Un unico ambiente che consente di fattorizzare I propri investimenti in know-how e in sviluppo

Windows Workflow Foundation **Parte Seconda**

Windows Workflow Foundation: il Runtime



Windows Workflow Foundation

The workflow framework and tools for Microsoft products and partner/customer ecosystem

- Unica tecnologia Workflow per il mondo Microsoft® Windows®
- Un framework per creare workflow, non per creare applicazioni o server
- Trasformare il modello dichiarativo dei workflow in un paradigma consolidato e facilmente assimilabile

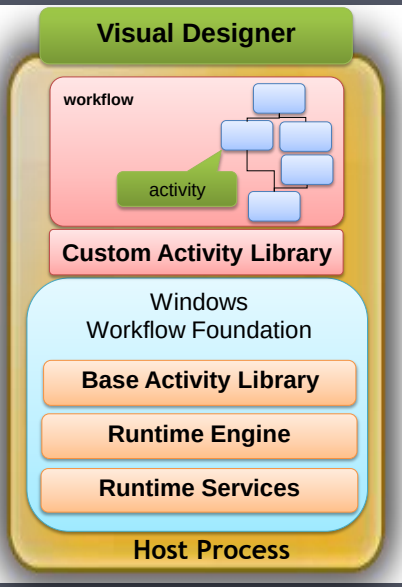
Windows Workflow Foundation Architecture

Key Concepts

- i Workflow sono un set di Activity
- un Workflow gira all'interno di un Processo Host, ovvero un'applicazione o un server
- gli sviluppatori possono realizzare le proprie Activity Libraries

Components

- Base Activity Library: activity atomiche utilizzabili inoltre per creare activity custom.
- Runtime Engine: controlla l'esecuzione del Workflow e la gestione dello stato relativo
- Runtime Services: Si occupa dell'Hosting e della comunicazione con il resto dei componenti
- Visual Designer: composizione grafica o code-based del workflow

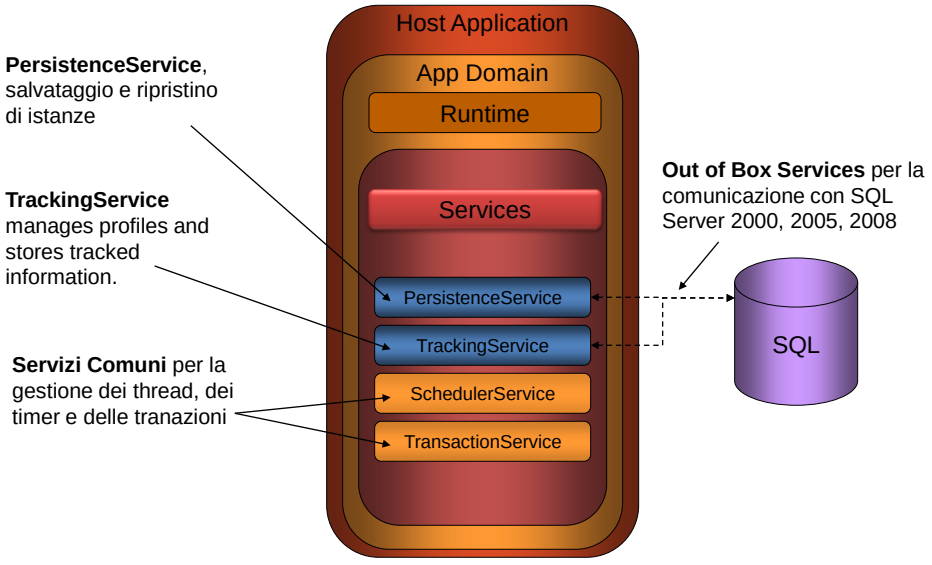


Windows Workflow Foundation

A workflow runtime is responsible for scheduling the execution of the steps

DotNet Campania team

Windows Workflow Foundation Runtime Unleashed



DotNet Campania team

Windows Workflow Foundation Runtime

Runtime Engine

- Il "core" del Workflow Runtime
- Responsabile dell'esecuzione e del ciclo di vita del workflow

Runtime Services

- Sono i servizi che permettono di interagire con le feauture di supporto ai workflow:
  - Persistenza
  - Tracking
  - Scheduling
  - …
- I servizi built-in possono essere sostituiti con dei custom runtime services, per esempio:
  - Con un *Custom OleDbPersistenceService* che implementa l'interfaccia IPersistenceService è possibile slavare lo stato di un workflow all'interno di un OleDB (di default è usato SQL Server)

DotNet Campania team

Windows Workflow Foundation Runtime

Workflow Runtime Properties	
Property	**Purpose**
IsStarted	Used to determine whether the workflow runtime has been started and is ready to accept workflow instances. *IsStarted* is *false* until the host calls *StartRuntime*. It remains *true* until the host calls *StopRuntime*.
Name	Gets or sets the name associated with the *WorkflowRuntime*. You cannot set *Name* while the workflow runtime is running (that is, when *IsStarted* is *true*). Any attempt to do so will result in an *InvalidOperationException*.
Workflow Runtime Methods	
AddService	Adds the specified service to the workflow runtime. There are limitations regarding what services can be added as well as when.
CreateWorkflow	Creates a workflow instance, including any specified (but optional) parameters. If the workflow runtime has not been started, the *CreateWorkflow* method calls *StartRuntime*.
GetWorkflow	Retrieves the workflow instance that has the specified workflow instance identifier (which consists of a Guid). If the workflow instance was idled and persisted, it will be reloaded and executed.
StartRuntime	Starts the workflow runtime and the workflow runtime services and then raises the *Started* event.
StopRuntime	Stops the workflow runtime and the runtime services and then raises the *Stopped* event.

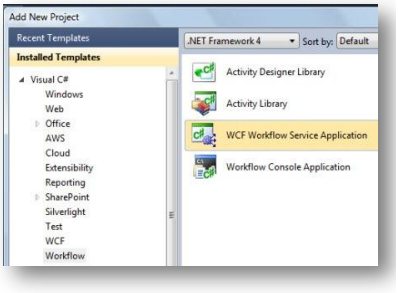
DotNet Campania team

Windows Workflow Foundation Runtime

Events	Purpose
Started	Raised when the workflow runtime is started.
Stopped	Raised when the workflow runtime is stopped.
WorkflowCompleted	Raised when a workflow instance has completed.
WorkflowIdled	Raised when a workflow instance enters the idle state. When workflow instances go idle, you have the opportunity to unload them from memory, store them in a database (in case they're waiting for a long-running task), and bring them back into memory at a later time.
WorkflowTerminated	Raised when a workflow instance is terminated. The workflow can be terminated by the host through a call to the *Terminate* method of a workflow instance, by a *Terminate* activity, or by the workflow runtime when an unhandled exception occurs.

Visual Studio e WFF

- Editing
  - Windows Workflow Designer
  - Tools per WF Design



- Templates
  - Workflow Activity Library
  - Simple Sequential Workflow Library/Console Application
  - Sharepoint 2007 Application

WF: Ma cos'è a livello di codice .NET?

Un workflow può essere realizzato in modo imperativo, ovvero dichiarando una classe

```
using System.Workflow.Activities;
public class MyWorkflow: SequentialWorkflow
{
  …
}
```

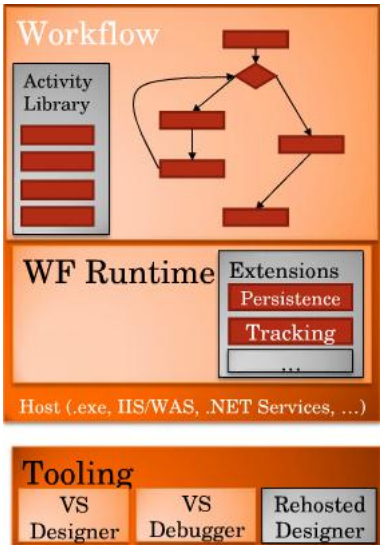
ma può anche essere «dichiarativo» tramite XML:

```
<?Mapping XmlNamespace="Activities"
ClrNamespace="System.Workflow.Activities"
Assembly="System.Workflow.Activities" ?>

<SequentialWorkflow x:Class="MyWorkflow" xmlns="Activities"
xmlns:x="Definition">
    …
</SequentialWorkflow>
```

DotNet Campania team

WF is… ART!

- **A**ctivities
- **R**untime
- **T**ooling



DotNet Campania team

Windows Workflow Foundation 4

- Workflow basati sul markup XAML(x) [soluzione preferita]
- Una libreria estesa di Activities di base
- Modello di sviluppo semplificato
  - Gestione delle istanze (Host)
  - Gestione dei Bookmark
  - Creazione di Activity Custom
  - Gestione dei dati
- Supporto per *Argomenti, Variabili ed Espressioni*
- Integrazione completa (fusione dei due team!) con WCF
- Runtime e Designer decisamente migliorati
- Histing e Management tramite AppFabric (Dublin)

DotNet Campania team

Windows Workflow Foundation 4: Data

Argomenti

- Usati per definire il modo in cui dati fluiscono all'interno di una Activity (In, Out e InOut)

Variabili

- Usate per gestire dati all'interno delle Activity

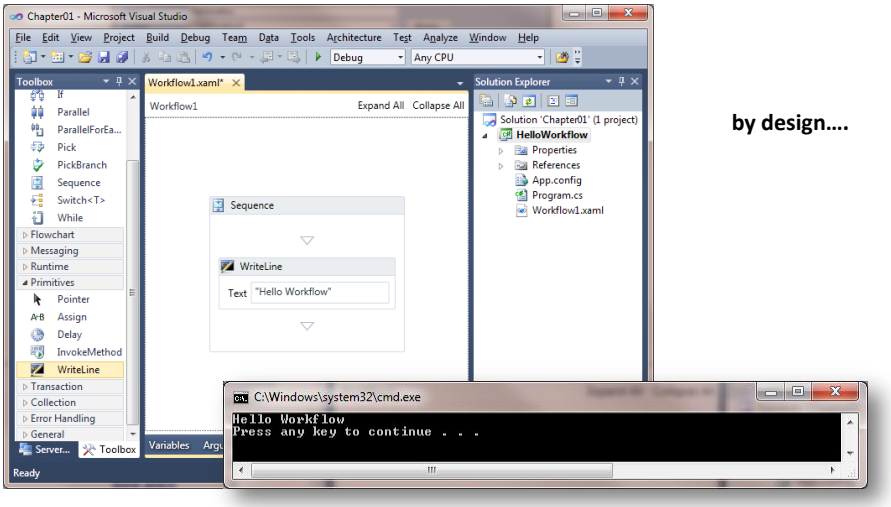
Espressioni

- Utilizzare per effettuare semplici operazioni sugli argomenti

DotNet Campania team

Hello Workflow

L'immancabile Hello Workflow… oops… era Hello World ☺



by design….

Hello Workflow

L'immancabile Hello Workflow… oops… era Hello World ☺

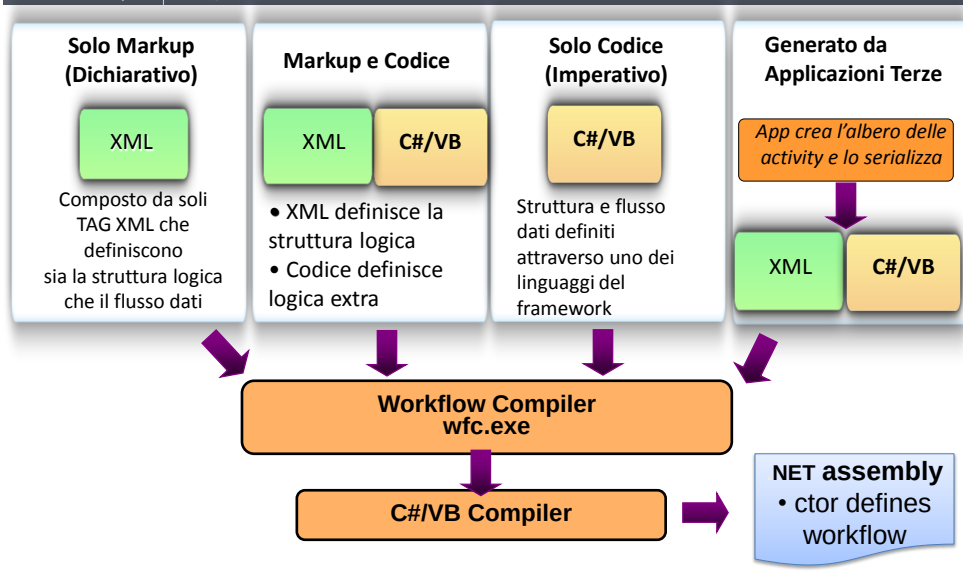
by code….

```
namespace HelloCodeWorkflow {
        class Program {
                static void Main(string[] args) {
                        WorkflowInvoker.Invoke(new HelloWorkflow());
                }
                }

public class HelloWorkflow:Activity {
        public HelloWorkflow() {
        this.Implementation = () => new Sequence {
                Activities = {
                        new WriteLine(){Text="Hello Workflow"}
                        }
                };
        }
    }
}
```


Parte Seconda

I Workflow nel mondo .NET:
IMPERATIVE CODE vs. DECLARATIVE DESIGN

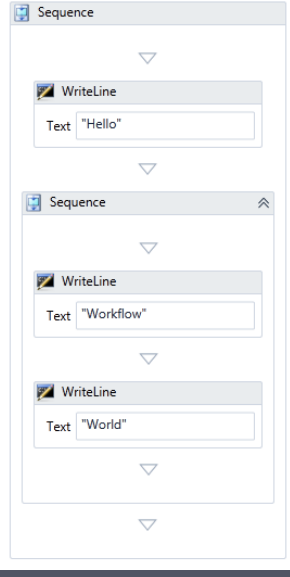
WF: Un modello misto, dichiarativo e imperativo



WF: modello Dichiarativo vs. modello Imperativo

WFManager.cs

```
1  using System.Activities;
2  using System.Activities.Statements;
3  public class WFManager{
4      public Activity GetWF() {
5          Sequence s = new Sequence() {
6              Activities ={
7                  new WriteLine(){Text="Hello"},
8                  new Sequence{
9                      Activities={
10                         new WriteLine{Text="Workflow"},
11                         new WriteLine{Text="World"}
12                     }
13                 }
14             }
15         };
16         return s;
17     }
18 }
```



WF: modello Dichiarativo vs. modello Imperativo

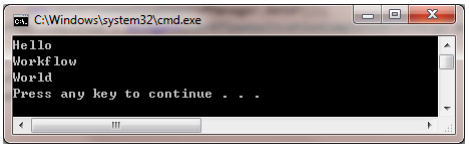
WFManager.cs

```
1  using System.Activities;
2  using System.Activities.Statements;
3  public class WFManager{
4      public Activity GetWF() {
5          Sequence s = new Sequence() {
6              Activities ={
7                  new WriteLine(){Text="Hello"},
8                  new Sequence{
9                      Activities={
10                         new WriteLine{Text="Workflow"},
11                         new WriteLine{Text="World"}
12                     }
13                 }
14             }
15         };
16         return s;
17     }
18 }
```

WF: modello Dichiarativo vs. modello Imperativo

Program.cs

```
1  using System;
2  using System.Activities;
3  using System.Activities.Statements;
4  using System.Activities.XamlIntegration;
5  using System.Collections.Generic;
6  using System.Threading;
7  class Program {
8      static void Main(string[] args) {
9          WFManager wfManager = new WFManager();
10         AutoResetEvent waitHandler = new AutoResetEvent(false);
11         WorkflowApplication wfApp =
12             new WorkflowApplication(wfManager.GetWF());
13         wfApp.Completed = delegate(WorkflowApplicationCompletedEventArgs arg) {
14             waitHandler.Set();
15         };
16         wfApp.Run();
17         waitHandler.WaitOne();
18     }
19 }
```



Windows Workflow Foundation **Parte Seconda**

WF: modello Dichiarativo vs. modello Imperativo

Sequence activity

```
public static Activity GetWF() {
    Variable<string> var =  new Variable<string>("var", "hello workflow");
    return new Sequence {
      Variables = { var },
      Activities ={
        new WriteLine{Text="Workflow Started"},
        new WriteLine{Text=new InArgument<string>(var)},
        new WriteLine{Text="Workflow Ended"}
      }
    };
  }
```


WF: modello Dichiarativo vs. modello Imperativo

Code Workflow

```
public class MyCodeWorkflow : Activity {
    public InArgument<string> inMSG { get; set; }
    public OutArgument<string> outMSG { get; set; }
    public MyCodeWorkflow() {
      this.Implementation = () => new Sequence {
        Activities = {
          new WriteLine{
            Text=new InArgument<string>(
              (activityContext)=>this.inMSG.Get(activityContext)
            )
          },
          new Assign<string>{
            To=new ArgumentReference<string>("outMSG"),
            Value=new InArgument<string>(
              (activityContext)=>this.inMSG.Get(activityContext)
            )
          } }};} }
  //host
  static void Main(string[] args) {
    IDictionary<string, object> input = new Dictionary<string, object>();
    input.Add("inMSG","hello");
    IDictionary<string, object> output = new Dictionary<string, object>();
    MyCodeWorkflow activity = new MyCodeWorkflow();
    output = WorkflowInvoker.Invoke(activity,input);
    Console.WriteLine(output["outMSG"]);
  }
```

Windows Workflow Foundation **Parte Seconda**

WF: modello Dichiarativo vs. modello Imperativo

Dynamic Activity

Permette di creare una nuova istanza di workflow a runtime senza la necessità di definire direttamente il wf.

```
public static DynamicActivity GetWF()
{
        return new DynamicActivity()
        {
                Implementation = () => new   Sequence()
                {
                        Activities  = {new WriteLine(){Text="Hello Workflow"} }
                }
        };
}
```


WF: modello Dichiarativo vs. modello Imperativo

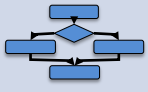
Da Codice (C#) a Desisgner (XAMLX)

Chi ha deciso di sviluppare il proprio workflow (WF4) direttamente da codice, può convertire rapidamente il proprio lavoro in XAMLX in modo da vederlo nel designer.
L'istruzione da usare è la seguente: XamlServices.Save(@"..\..\demo.xaml", workflow), dove workflow è definito come, ad esempio, nello spezzone di codice seguente:

```
var workflow = new Sequence();
workflow.Activities.Add(new WriteLine() { Text = "Hello workflow." });
workflow.Activities.Add(new Persist());
workflow.Activities.Add(new If()
{
Condition = new VisualBasicValue<bool>("System.DateTime.Now.Hour < 12"),
Then = new WriteLine() { Text = "Good morning" },
Else = new WriteLine() { Text = "Good afternoon" }
});
workflow.Activities.Add(new WriteLine()
{
Text = new VisualBasicValue<string>("\"The current time is: \" &
System.DateTime.Now.ToLongTimeString()")
});
```

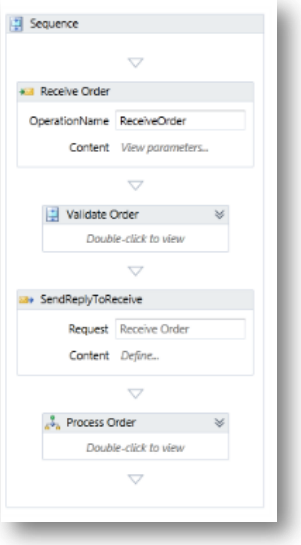
Windows Workflow Foundation **Parte Seconda**

I Workflow nel mondo .NET: TIPOLOGIE DI WORKFLOW

Type	Meaning
Sequenziali 	I task sono eseguiti autonomamente con la possibilità di piccole variazioni del path. La Root Activity di riferimento è *SequentialWorkflow*.
State-machine, similare ad un grafo 	I task del Workflow dipendono da eventi esterni che ne guidano l'evoluzione. La Root Activity di riferimento è *StateMachineWorkflow.*
Flowchart	Pensati per il mapping 1:1 dei flowchart di progetto. La Root Activity di riferimento è *Flowchart*
Rules-based	Pensati per risolvere problemi elevata complessità che non possono essere risolti con i modelli precedenti. I Rules-based workflow possono avere come Root Activity indistintamente quelle precedenti.

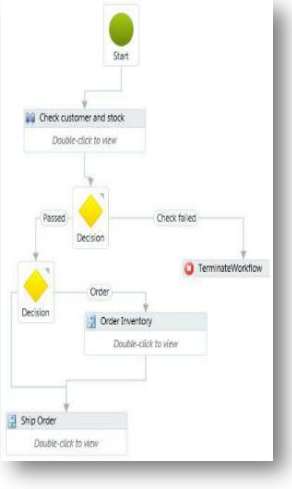
Workflow Types: Sequential Workflow

- Caratteristiche
  - Ben Strutturato
  - Esecuzione Esplicita
- Limitazioni
  - Non è possibile ripetere l'esecuzione di un'activity precedente senza un loop;
  - Non si possono avere due loop di ritorno intersecati;
  - Poco flessibile nel caso gli eventi esterni che condiziona il flusso non possono essere garantiti nell'ordine stabilito;
- Scenari
  - Automazione, Processi di documentazione
  - Processi rigidi



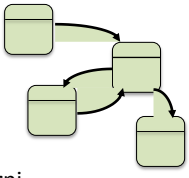
Workflow Types: FlowChart

- Caratteristiche
  - Particolarmente Flessibile
  - Esecuzione gestita dai collegamenti
  - Possibilità di «saltare» ovunque
- Limitazioni
  - La descrizione del flusso può diventare confusionale
  - Si possono creare situazioni di «jump» non desiderati
- Scenari
  - Processi di interazione umana



Workflow Types: State Machine

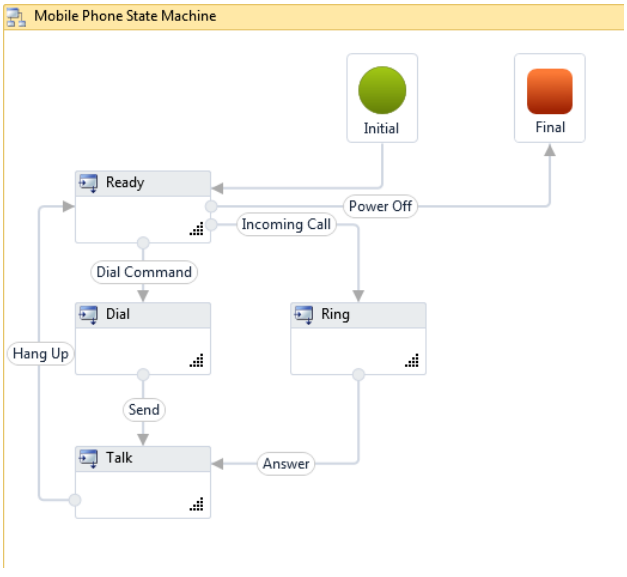
- Caratteristiche
  - *Event driven*
    - pensato per gli scenari «system to system»,
    - o «human to system» indiretti (via application)
  - Esecuzione basata sullo Stato corrente
  - Attesa per le transazioni
  - Maggiormente flessibile rispetto a cambiamenti esterni
  - Inizia con lo stato di «INIT» e termina con quello di «COMPLETED»



- Limitazioni
  - Non è possibile creare activity con stato riutilizzabili
  - Il Designer e il Debugger non supportano pienamente il modello

- Scenari
  - Processi di approvazione a livelli multipli

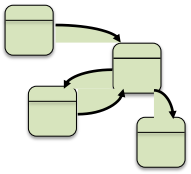
Workflow Types: State Machine



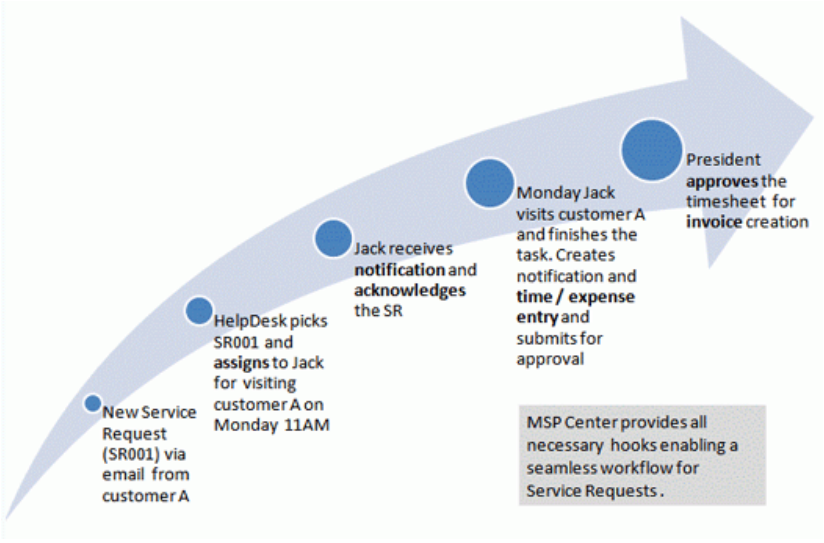
Windows Workflow Foundation **Parte Seconda**

State Machine Workflow: il dilemma di casa Redmond

- In realtà le State Machine sono state rimosse dalla versione 4.0 del dotNet Framework, o meglio è stato rimosso il relativo template da VS, poiché è sostanzialmente possibile ottenere lo stesso risultato con un FlowChart
- Successivamente MS ha rilasciato lo stesso su CodePlex vista la relativa comodità.
- Infine con .NET vNext (alias .Net 4.5) le State Machine verranno ufficialmente reintrodotte.



Esempio di processo di approvazione



DotNet Campania team

Demo: Simple Workflow

Hello World Workflow

- Creazione di un Workflow con l'utilizzo di Activities Primitive
- Creazione di una Console Application
  - Istanza del Workflow Runtime
  - Istanza dell'HelloWorldWorkflow
  - Avvio del Workflow

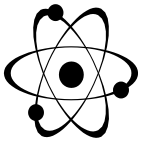
DotNet Campania team

Parte Seconda

I Workflow nel mondo .NET:
ACTIVTIES

A workflow is composed of an ordered sequence of activities

DotNet Campania team
Windows Workflow Foundation: Activites



An activity is the atom in the workflow universe.

A workflow has data which is captured as variables

DotNet Campania team
Windows Workflow Foundation: Activites

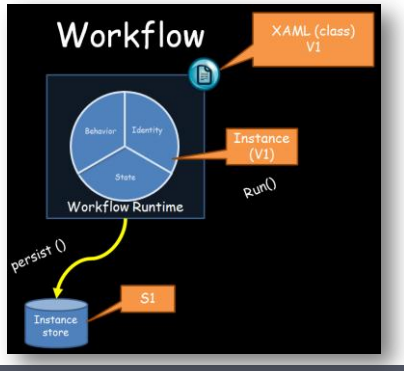
An activity has arguments to allow data to flow in and out

Activites: i mattoncini lego di WWF

- Le Activity sono i “mattoncini” (work item) portati dei Workflow, ovvero un’unità minimale di lavoro
- Activity = Data + Work
  - Data
    - Argomenti: passaggio di informazioni
    - Variabili: elementi di storage per l’elaborazione
  - Work
    - Codice con logica applicativa
      - Proprio
      - Riuso
- Una Activity può contenere altre Activity.
- Dispone di Parametri di Input/Output
- Dispone di variabili interne

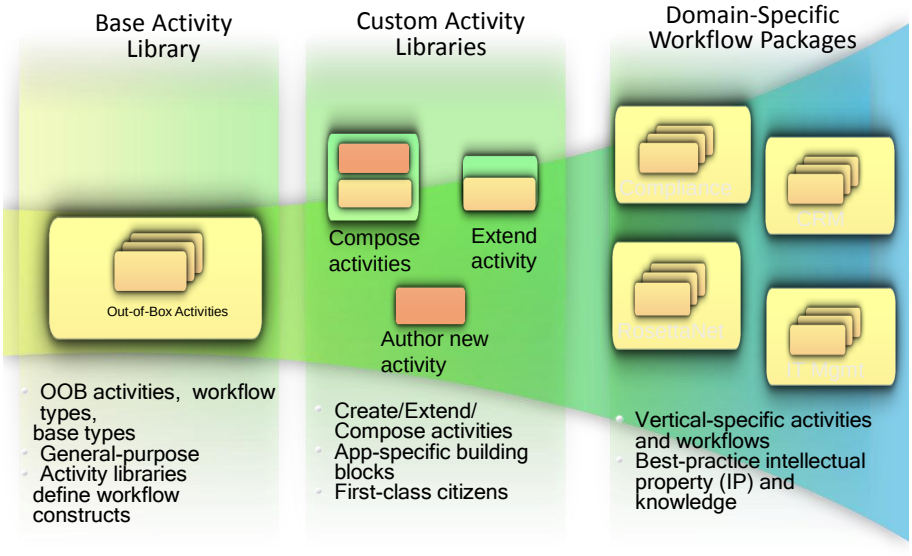
Anche il Workflow stesso è un Activity!

- Un WF è a sua volta un activity (possiamo immaginarlo come la Root Activity)
- Un WF rappresenta la “definizione” di un programma, di cui è possibile creare più istanze.
- Ogni istanza ha un ambiente
- (context) specifico



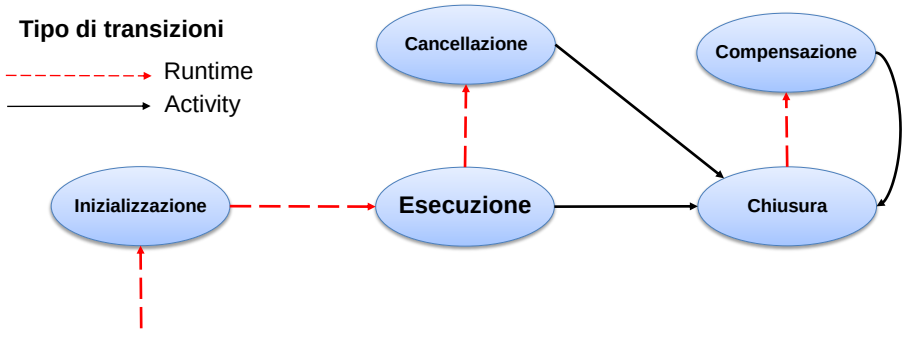
DotNet Campania team

Activites: un approccio estendibile



DotNet Campania team

Activity: Stati



Activity Execution Methods

- Initialize()
- **Execute()**
- Cancel()
- Compensate()

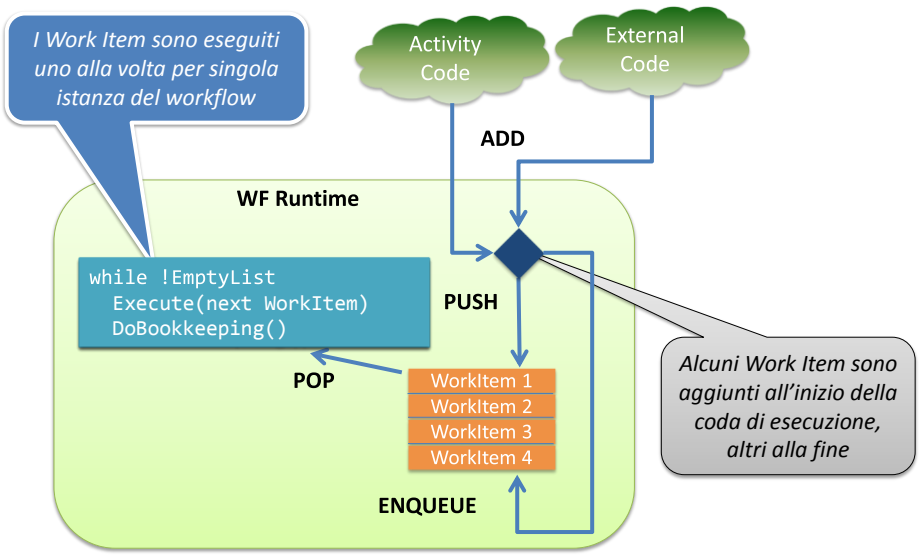
ActivityExecutionStatus Enumeration

- Initialized
- Executing
- Compensating
- Cancelling
- Closed
- Faulting

Activites: esecuzione

- Il runtime di WF lavora e conosce solo *activity*, *activity* e null'altro che *activity*
  - non entra nello specifico tipo (not Sequence, Parallel, Recurrence)
  - WF si comporta come un "arbitro" che impone le regole del gioco (esecuzione)
  - Attraverso il metodo **CacheMetadata** un'activity descrive se stessa;
- Attraverso gli stati prima descritti, il runtime è in grado di monitorare step-by-step l'esecuzione dell'activity
- Un activity può schedulare l'esecuzione di una o più activities figlie, ed essere informata sul relativo stato
- Percorsi (path) multipli di esecuzione

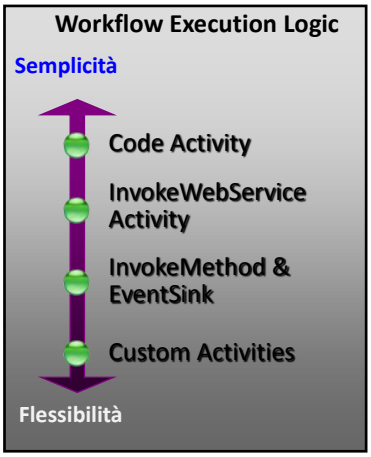
Activites: schedulazione



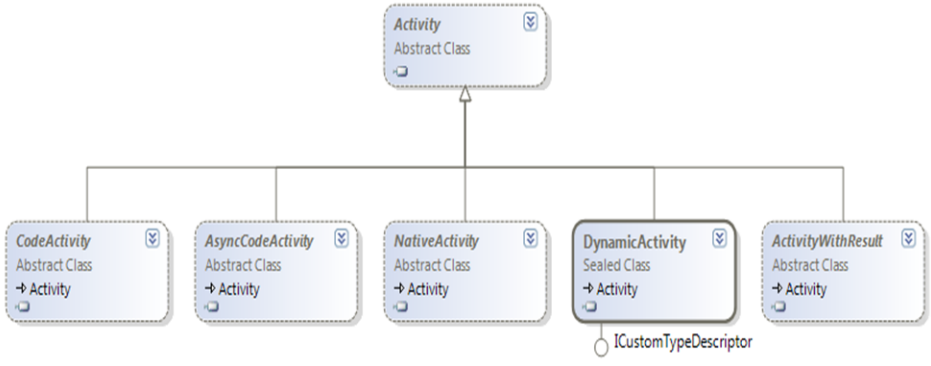
Activites: Custom Activities

In sintesi un Activity è un'unità che:

- *Permette l'esecuzione di uno specifico compito*
- *Favorisce il riuso*
- *Favorisce la "composizione"*
- *E' il mattone dei WF*



Activites: Namespace System.Activities



DotNet Campania team

Activites: Proprietà

Property	Purpose
Description	Gets or sets the user-defined description of the activity.
Enabled	Gets or sets a value that indicates whether this instance is enabled for execution and validation.
ExecutionResult	Gets the *ActivityExecutionResult* of the last attempt to run this instance (*Canceled*, *Compensated*, *Faulted*, *None*, and *Succeeded*).
ExecutionStatus	Gets the status of the workflow in the form of one of the
ActivityExecutionStatus	values (*Canceling*, *Closed*, *Compensating*, *Executing*, *Faulting*, and *Initialized*).
Name	Gets or sets the name of this activity instance.
Parent	Gets the activity that encloses this activity.
WorkflowInstanceId	Gets the workflow instance identifier associated with this activity.

DotNet Campania team

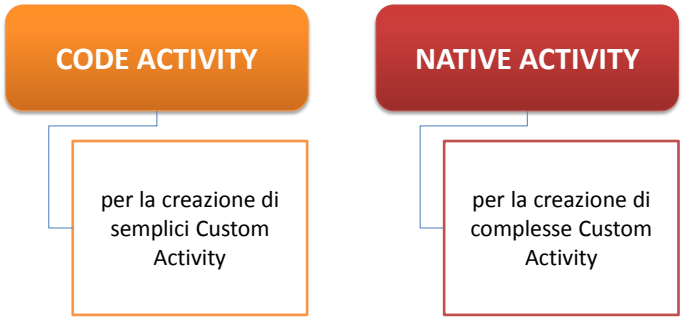
Activites: Metodi

Method	Purpose
Cancel	Cancels the execution of an activity.
Clone	Returns a deep copy of the activity (“deep copy” means the clone contains all of the internal data from the cloned activity).
Execute	Synchronously runs the activity.
GetActivityByName	If executed on a composite activity, this method returns the named activity if it is contained by the composite activity.
Load	Loads an instance of an activity from a stream.
RaiseEvent	Raises an event associated with the specified *DependencyProperty*.
RaiseGenericEvent<T>	Raises the event associated with the referenced *DependencyProperty*. The effect of *RaiseEvent* and *RaiseGenericEvent* is the same—fire an event. *RaiseEvent* uses the dependency property directly, while *RaiseGenericEvent* is the generic (templated) version.
Save	Saves a copy of the activity to a stream.

Activites: Activity Composition

- Creare una nuova Activity partendo da Activities esistenti (COMPOSIZIONE). E' possibile ottenere questo risultato in modo:
  - **dichiarativo**;
  - **compilando o eseguendo activity direttamente tramite DynamicActivity**
- Da preferire quando…
  - …si sta creando un nuovo layer di astrazione;
  - …si sta creando un activity in cui si intende riutilizzare il lavoro precedente;
- Non usare quando…
  - …si ha la necessità di un controllo granulare sull'attività che si vuole realizzare.

CodeActivity (high level) vs. NativeActivity (lowest level)



Una **CustomActivity** estende WF creando un'unità atomica di esecuzione.

Activites: CodeActivity (high level)

- E' sufficiente effettuare l'override del metodo .Execute() e definire i parametri di input/outpu t come: **InArgument**, **OutArgument** oppure **InOutArgument**
- Istanze multiple e relativo «context» per singola definizione di WF
  - argomento.Get(context)
  - context.GetValue(this.Argomento)

Da usare quando…

- …si ha del codice di business da «incapsulare» nel mondo WF;
- …per creare velocemente un Activity Custom con del codice tradizionale;

Non usare quando…

- …è necessario schedulare l'esecuzione di altre Activituies (è necessario usare la classe NativeActivity)
- …quando si vuole ottenere un'esecuzione asincrona, essendo la CodeActivity bloccante (usare AsyncCodeActivity)

Activites: CodeActivity (high level)

```
public class DoubleActivity : CodeActivity
{
        public InArgument<int> Value {get; set;}
        public OutArgument<int> Double {get; set;}

        protected ovveride void Execute(CodeActivityContext context)
        {
                Double.Set(context, Value.Get(context) * 2);
        }
}
```

```
public class DoubleActivity : CodeActivity<int>
{
        public InArgument<int> Value {get; set;}

        protected ovveride int Execute(CodeActivityContext context)
        {
                return Value.Get(context) * 2;
        }
}
```

Windows Workflow Foundation **Parte Seconda**

Activites: Native Activity (lowest level)

- Consente maggiore flessibilità e maggiore controllo;
- Può essere **Sincrona** o **Asincrona** (tramite la creazione di Bookmark)
  - Context.CreateBookmark(«MyBookmark», MyCallback);
- Può schedulare Activities figlie
  - Context.ScheduleActivity();
- Termina quando tutte le Activities figlie ed i Bookmark sono terminati;

Da usare quando…

- …è necessario un interazione diretta con il WF Runtime
- …si vuole realizzare un'Activity in grado di schedulare Activities figlie;
- …si vogliono creare bookmarks per un punto di Resume;

Non usare quando…

- …tutto il codice può essere facilmente contenuto e gestito nel metodo .Execute(); (usare una CodeActivity)

Fare attenzione a…

- Gestione degli errori, cancellazione
- Gestione Esplicita e Implicita delle Activities figlie

Activites: Native Activity (lowest level)

```
public class ReadString: NativeActivity
{
        public OutArgument<string> Name {get; set;}
        protected override CanInduceIdle {get; {return true;} }

        protected ovveride void Execute(ActivityExecutionContext context)
        {
                context.CreateNamedBookmark(«input», new
                        BookmarkCallback(this.OnBookmarkCallback));
        }

        private void OnBookmarkCallback(ActivityExecutionContext context, Bookmark
                bookmark, object obj)
        {
                this.Name.Set(context, (string), obj);
        }
}
```

```
workflowApp.Run()
string text = Console.ReadLine();
workflowApp.ResumeBookmark(«input», text,);
syncEvent.WaitOne();
```

Windows Workflow Foundation **Parte Seconda**

Activites: ActivityDelegate

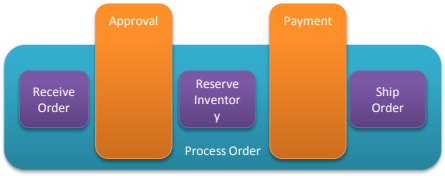
- Permette al “consumer” di iniettare le proprie funzionalità
- Concettualmente equivalente alle *Func*, *Action* o ai Predicate

Da usare quando…

- …si desidera creare un place-holder;
- …il consumer deve poter modificare parte del comportamento di esecuzione dell’activity

Non usare quando…

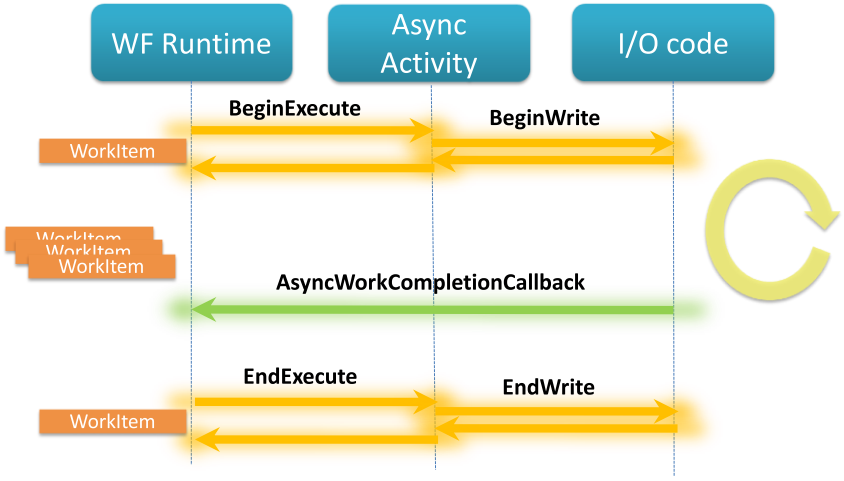
- …non è richiesta generalizzazione
- … al posto della «composizione»



Parte Seconda

I Workflow nel mondo .NET:
ACTIVITIES ASICONCRONE

Activities Asincrone



Activities Asincrone

Parallel + async activities = true concurrency

```
Parallel p = new Parallel
{
   Branches =
   {
      new WriteToFile { FileName = "a", Bytes = … },
      new WriteToFile { FileName = "b", Bytes = … }
   }
};
WorkflowInvoker.Invoke(p);
```

DotNet Campania team

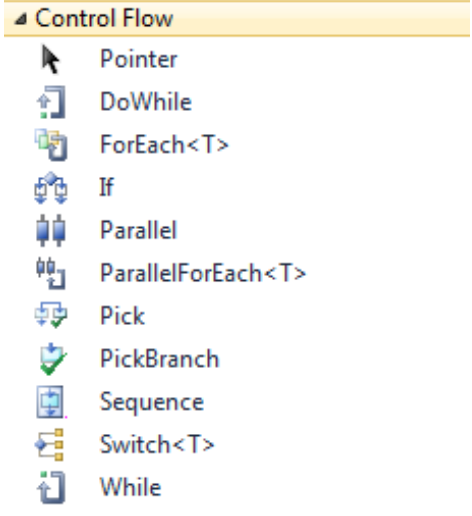
Activities

I Workflow nel mondo .NET:
BUILT-IN ACTIVITIES

DotNet Campania team

Built-in Activities: Control flow

- **DoWhile**
- **ForEach<T>**
- **If**
- **Parallel**
- **ParallelForEach<T>**
- **Pick**
- **PickBranch**
- **Sequence**
- **Switch<T>**
- **While**



Built-in Activities: Control flow

Attenzione al concetto di parallelismo in WF4:

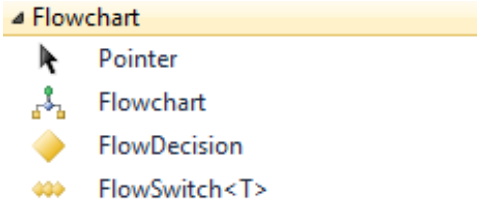
l'activity Parallel non avvia i relativi rami in parallelo (tramite thread separati), ma li esegue entrambi all'interno dello stesso contesto di esecuzione, sostanzialmente accodandoli.

Built-in Activities: Flowchart

- **FlowChart**

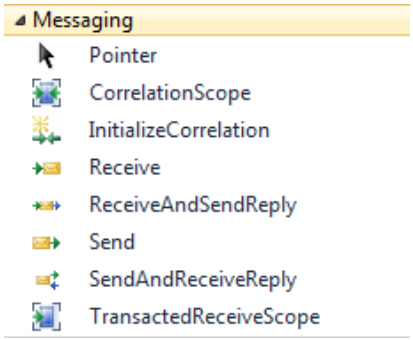
  Crea un nuovo flowchart
- **FlowDecision**

  Inserisce un punto di switch
- **FlowSwitch<T>**
- Inserisce un punto di switch generico



Built-in Activities: Messaging

- **CorrelationScope**
  Definisce lo Scope (area) in cui è valido il contesto di correlazione, sfruttando l'handler di correlazione specifico
- **InitializeCorrelation**
  Inizializza il contesto di correlazione
- **Receive**
  Espone un entry-point per la ricezione di un messaggio. Assimilabile ad un operation di tipo OneWay WCF
- **ReceiveAndSendReply**
  Espone un entry-point per la ricezione di un messaggio e l'invio di una risposta. Assimilabile ad un operation WCF
- **Send**
  Invio di un messaggio ad un servizio Web senza attesa della risposta
- **SendAndReceiveReply**
  Invio di un messaggio ad un servizio Web con attesa della risposta
- **TransactedReceiveScope**
  Consente di eseguire il flusso all'interno del context creato alla ricezione del messaggio



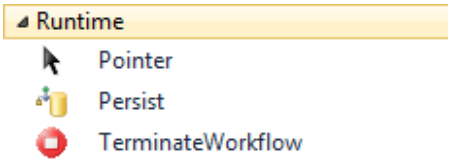
Built-in Activities: Messaging, Known Type

Se si prova ad impostare una *Receive* un message type root di una gerarchia di oggetti (nell'es. precedente: *Forma*), bisogna poi utilizzare proprio un oggetto Forma e non uno dei suoi figli. Neanche il downcasting aggira questo problema perché il messaggio ottenuto dalla serializzazione dello stesso ha un elemento di Root diverso che porta la Receive a scartarlo.

L'unica soluzione trova è quella di avere uno Switch con N. Receive/SendReplay.
La cosa si applica anche alla Send/ReceiveReplay

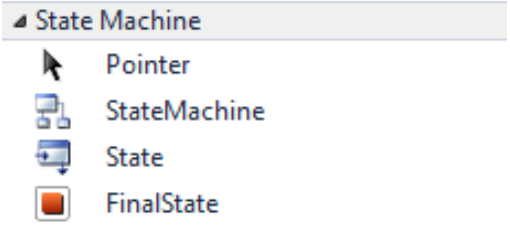
Built-in Activities: Runtime

- **Persist**
  - Permette di persiste in modo deterministico il workflow, salavandolo nel repository scelto
- **TerminateWorkflow**
  - Inserisce un punto di terminazione “forzata” nel flusso



Built-in Activities: State Machine*

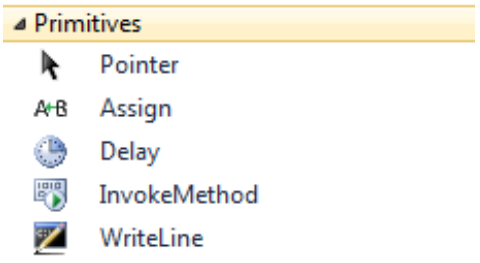
- **StateMachine**
  - Crea un WF di tipo StateMachine
- **State**
  - Aggiunge uno stato
- **FinalStake**
  - Aggiunge lo Stato finale



*Presente in WF3.5, in WF4.0 bisogna scaricare l’estensione da Codeplex, reintrodotto in WF4.5

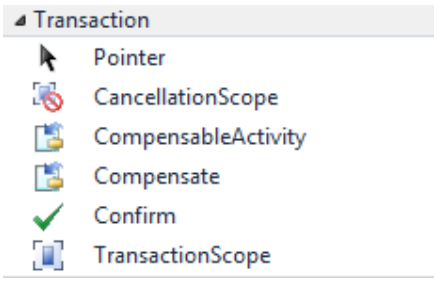
Built-in Activities: Primitives

- **Assign**
  - Assegnazioni di valori ad una variabile/ proprietà
- **Delay**
  - Inserisce un ritardo nel flusso
- **InvokeMethod**
  - Permette di invcare un metodo di una classe (statico o di istanza)
- **WriteLine**
  - Scrive una stringa sull'stdOut



Built-in Activities: Transaction

- **CancellationScope**
- **ComplensableActivity**
- **Compensate**
- **Confirm**
- **TransactionScope**

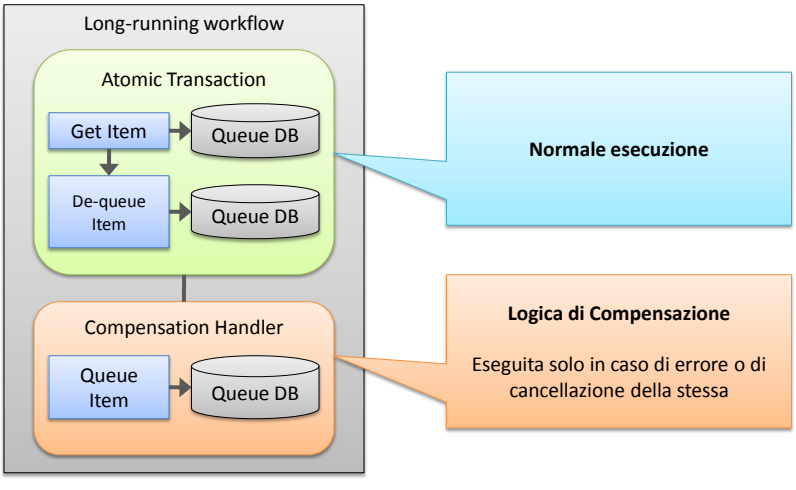


Transazioni in Windows Workflow

- Gestione delle Transazioni:
  - –Atomicità (ACID)
  - –Comportamento simile alla classe System.Transactions.TransactionScope
  - –Non possono essere innestate
  - –Non è possibile gestire le eccezioni

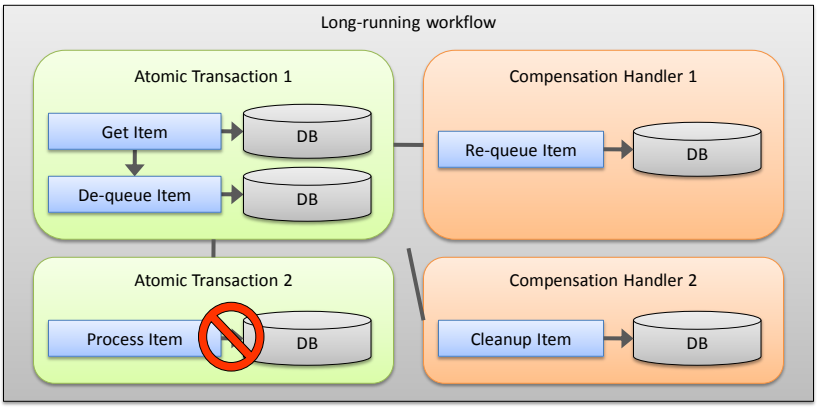
- Le transazioni non sono pensate per I processi long-running: non potete bloccare una tabella per 3 giorni!
- Pemettono la “Compensazione” nel caso in cui qualcosa non vada a buon fine o un’operazione venga annullata: ad esempio un prodotto deve essere re-inserito nell’inventario se l’acquirente cancella l’ordine.

Transazioni in Windows Workflow: Compensazione



DotNet Campania team

Compensating Flow

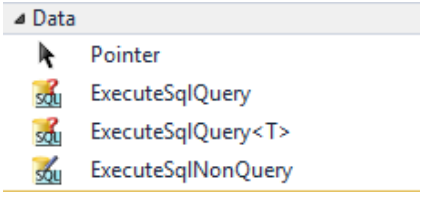


- If transaction 1 commits and transaction 2 fails then compensation handler 1 will be executed
- The item will be re-queued into the database

DotNet Campania team

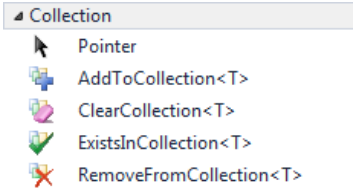
Built-in Activities: Data

- **ExecuteSqlQuery**
- **ExecuteSqlQuery<T>**
- **ExecuteSqlNonQuery**



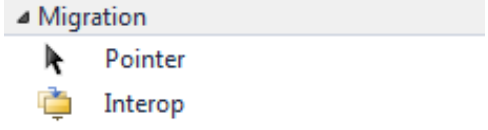
Built-in Activities: Collection

- **AddToCollection<T>**
- **ClearCollection<T>**
- **ExistsInCollection<T>**
- **RemoveFromCollection<T>**



Built-in Activities: Migration

- **Interop**



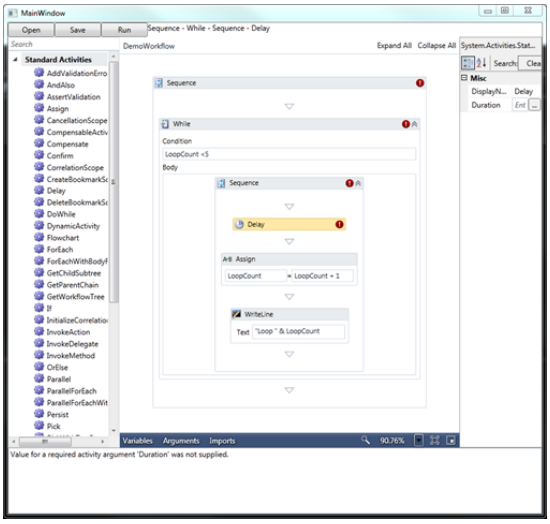
DEMO!

DotNet Campania team
Parte Seconda

I Workflow nel mondo .NET:
BUILT-IN ACTIVITIES - REHOSTING WORKFLOW DESIGNER

Rehosting the Workflow Designer in WF4

WWF 4 offre la possibilità di «inglobare» all'interno della propria applicazione WPF il Designer dei Workflow.



Rehosting the Workflow Designer in WF4

Le classi di riferimento sono:

- **ActivityDesigner**
- **WorkflowItemPresenter**
- **WorkflowItemsPresenter**
- **ExpressionTextBox**

Windows Workflow Foundation **Parte Seconda**

Rehosting the Workflow Designer in WF4

La classe di riferimento è **WorkflowDesigner**, che espone sia l'area di lavoro che le altre proprietà relative:

- *View,* design surface
- *PropertyInspectorView*

```
workflowDesigner = new WorkflowDesigner();
_workflowDesigner.Load(_fileName);

var view = _workflowDesigner.View;
Grid.SetColumn(view, 1);
Grid.SetRow(view, 1);
LayoutGrid.Children.Add(view);

var propInspector = _workflowDesigner.PropertyInspectorView;
Grid.SetColumn(propInspector, 2);
Grid.SetRow(propInspector, 1);
LayoutGrid.Children.Add(propInspector);
```

Lo step successivo è quello di registrare i metadata, ma è necessaria per utilizzare le activity in modo espanso.

```
new DesignerMetadata().Register();
```


Rehosting the Workflow Designer in WF4

… per salvare o caricare il worfklow basta utilizzare i metodi:

- **Load()**
- **Save()**

della solita classe WorkflowDesigner.

Windows Workflow Foundation **Parte Seconda**

DotNet Campania team

Rehosting the Workflow Designer in WF4

Ora, aggiungere la ToolBox al proprio progetto:

```
var toolbox = new ToolboxControl();
var cat = new ToolboxCategory("Standard Activities");
var assemblies = new list<Assembly>();
assemblies.Add(typeof(Send).Assembly);
assemblies.Add(typeof(Delay).Assembly);
assemblies.Add(typeof(ReceiveAndSendReplyFactory).Assembly);

var query =     from asm in assemblies
                from type in asm.GetTypes()
                where type.IsPublic &&
                !type.IsNested &&
                !type.IsAbstract &&
                !type.ContainsGenericParameters &&
                (typeof(Activity).IsAssignableFrom(type) ||
                typeof(IActivityTemplateFactory).IsAssignableFrom(type))
                orderby type.Name
                select new ToolboxItemWrapper(type);

query.ToList().ForEach(ti => cat.Add(ti));toolbox.Categories.Add(cat);
Grid.SetColumn(toolbox, 0);
Grid.SetRow(toolbox, 1);
LayoutGrid.Children.Add(toolbox);
```


DotNet Campania team

Rehosting the Workflow Designer in WF4

… aggiungiamo il codice per visualizzare l'Activity correntemente selezionata ed i relativi «figli», registrando l'handler:

```
_workflowDesigner.Context.Items.Subscribe<Selection>(SelectionChanged);
```

ed implementandone il codice:

```
private void SelectionChanged(Selection selection)
{
        var modelItem = selection.PrimarySelection;
        var sb = new StringBuilder();

        while (modelItem != null)
        {
                var displayName = modelItem.Properties["DisplayName"];
                if (displayName != null)
                {
                        if (sb.Length > 0) sb.Insert(0, " - ");
                        sb.Insert(0, displayName.ComputedValue);
                }
                modelItem = modelItem.Parent;
        }

        CurrentActivityName.Text = sb.ToString();
}
```

Windows Workflow Foundation **Parte Seconda**

Rehosting the Workflow Designer in WF4

… ed ora il codice per la validazione del workflow. Registriamo l'handler che cattura gli eventi di modifica:

```
var validationErrorService = new
ValidationErrorService(WorkflowErrors.Items);_workflowDesigner.Context.Servic
es.Publish<IValidationErrorService>(validationErrorService);
```

ed implementandone il codice (dando un corpo alla *IValidationErrorService*:

```
public class ValidationErrorService : IValidationErrorService
{
        private IList _errorList;
        public ValidationErrorService(IList errorList)
        { _errorList = errorList; }

        public void ShowValidationErrors(IList<ValidationErrorInfo> errors)
        {
                _errorList.Clear();
                foreach (var error in errors)
                { _errorList.Add(error.Message); }
        }
}
```


Rehosting the Workflow Designer in WF4

… eseguiamo il nostro applicativo!

```
var writer = new StringWriter();
var workflow = ActivityXamlServices.Load(_fileName);
var wa = new WorkflowApplication(workflow);
wa.Extensions.Add(writer);
wa.Completed = WorkflowCompleted;
wa.OnUnhandledException =
WorkflowUnhandledException;wa.Run();
```

Windows Workflow Foundation **Parte Seconda**

DEMO!
workflowdesigner_embedded.zip

DotNet Campania team
Parte Seconda

I Workflow nel mondo .NET:
ESECUZIONE DI UN WORKFLOW

Esecuzione di un Workflow

Per eseguire un Workflow si hanno essenzialmente tre possibilità:

- ***Workflow Invoker***
- ***Workflow Application***
- ***Workflow ServiceHost***

Esecuzione di un Workflow: WorkflowInvoker

WokflowInvoker

- ***semplice***
- ***sincrono***
- ***ottimo per test di unità (Unit Test)***
- ***viene eseguito nello scope del Thread chiamante***

```
var workflow = new HelloWorkflow();
workflow.DisplayName = «Hello Workflow»;

var output = WorkflowInvoker.Invoke(workflow);
PrintOutput(output);
```

Esecuzione di un Workflow: WorkflowApplication

WokflowApplication

- ***potente***
- ***asincrono***
- ***funzionalità callback per comunicare i cambiamenti di stato***
- ***utilizzo di SynchronizationContext per controllare il threading***

```
var workflow = new HelloWorkflow();
workflow.DisplayName = «Hello Workflow»;

var wa= new WorkflowApplication(workflow);
wa.Completed => PrintOutput(e.Outputs);
wa.Run();
```


Esecuzione di un Workflow: WorkflowServiceHost

WokflowServiceHost

- ***espone un workflow come un servizio WCF***
- ***può essere self-hosted***
- ***può essere gestito (hosting) da IIS/WAS (appFabric***

```
var workflow = new HelloWorkflowService();
var baseAddress = new Uri(«http://localhost/HelloWorkflow»);
var wsh = new WorkflowServiceHost(workflow, baseAddress);

wsh.Open();
Console.WriteLine(«The WorkflowSertvice is listening.»);
Console.ReadLine();
wsh.Close();
```

Torneremo successivamente ed approfonditamente sui Workflow Service

Dal punto di vista del funzionane a runtime, abbiamo due scenari possibili:

- **Hosted Workflow**
  - gestiti tramite un'app host (console, win Forms, ASP.NET, Windows Service, ecc)
- **Workflow Service** (introdotti con .net 3.5)
  - gestiti come un servizio WCF tramite IIS e AppFabric.

WORKFLOW SERVICES

WF Service: aWCF service whose implementation is a workflow

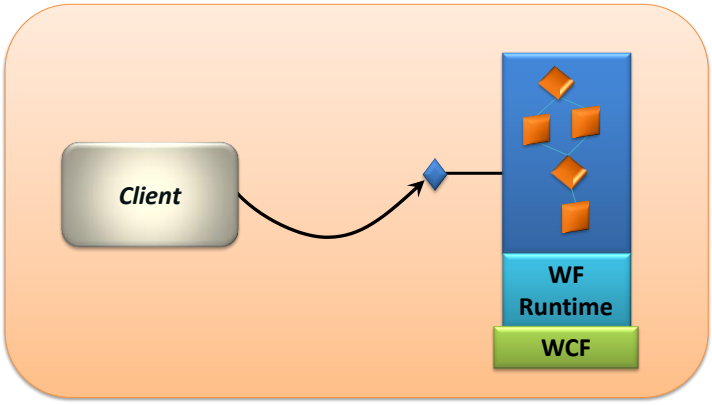
- I Workflow Services sono un «ibrido» nato dall'avvicinamento di WCF a WF, iniziato nel framework 3.5 e rafforzato (quasi fondendo insieme i due layer) nel 4.

WFC fuori e WF dentro

- WCF espone le interfacce verso l'esterno
- WF descrive il flusso (la logica) e gestisce i cambiamenti di stato

Workflow Service

"servizio WCF la cui logica è implementata tramite un Workflow"



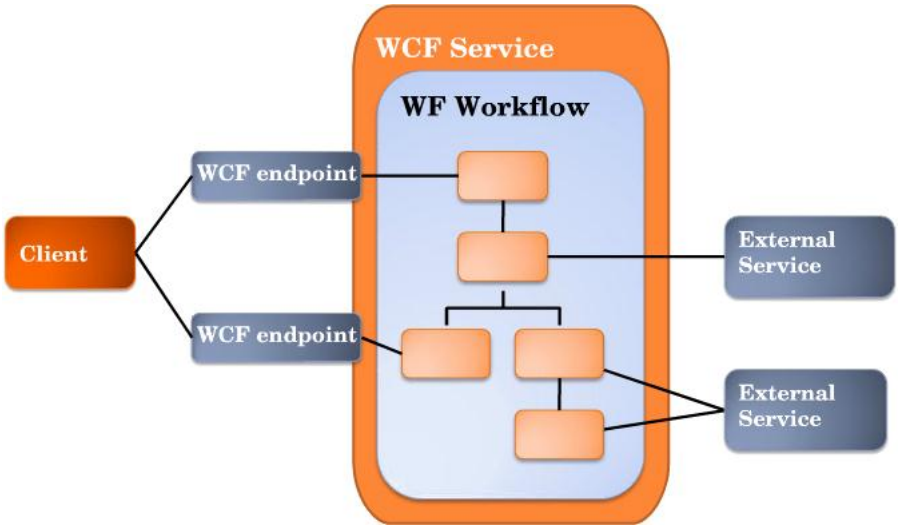
DotNet Campania team

Workflow Service

Service logic easily modeled as a workflow

Windows Workflow Foundation **Parte Terza**

Workflow Service



Workflow Service: un mondo di servizi… XAML

- Con i WF Service è possibile sostanzialmente definire un servizio attraverso il linguaggio di markup XAML
  - *declarative WCF workflow service*
  - *endpoint completamente definito in XAML*
  - *implementazione definita tramite XAML*
  - *l'intero servizio è definito in un unico file*
  - *specifiche activities pensate per il mondo dei servizi*

ACTIVITIES

- Send
- Receive
- SendReply
- ReceiveReply
- CorrelationScope
- InizializeCorrelation
- ….
- …

DotNet Campania team

Workflow Service: goals

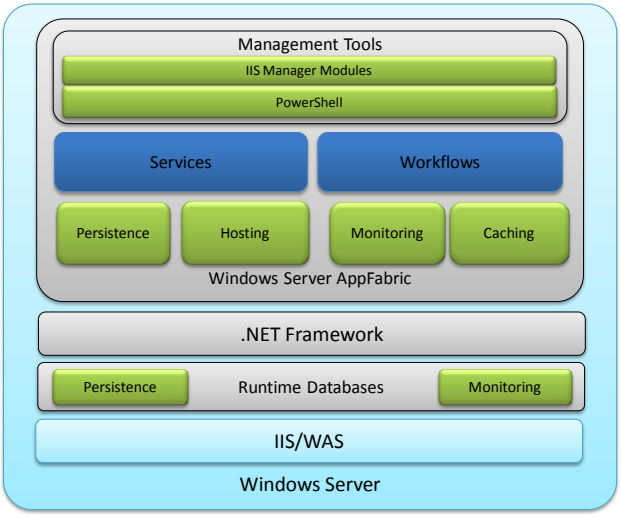
- I processi sono sempre più distribuiti e quindi la complessità viene sempre più gestita attraverso applicazioni service-oriented.
- La complessità è più gestibile se scomposta in processi ben definiti coordinabili tra loro
- Riutilizzare un environment service-oriented noto (IIS, AppFabric)
- Alta disponibilità di strumenti di monitoring

DotNet Campania team

Workflow Service

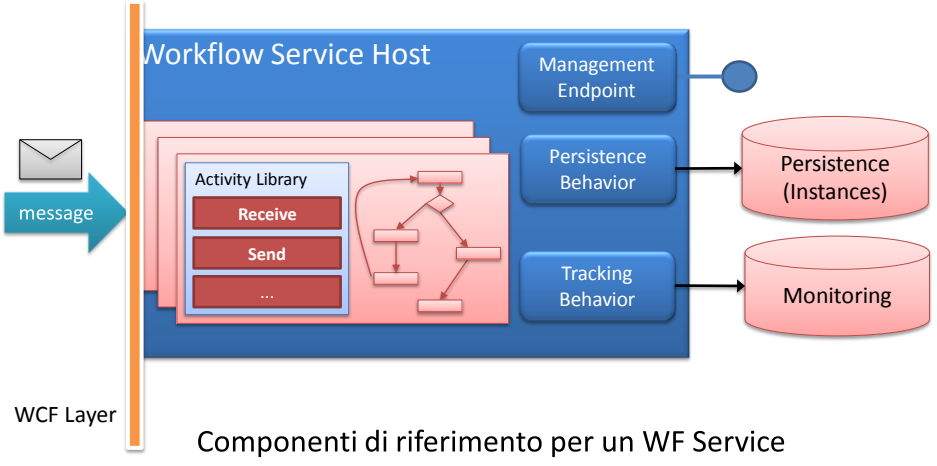
Service encapsulates a long running process

WCF E WF 4 Environment



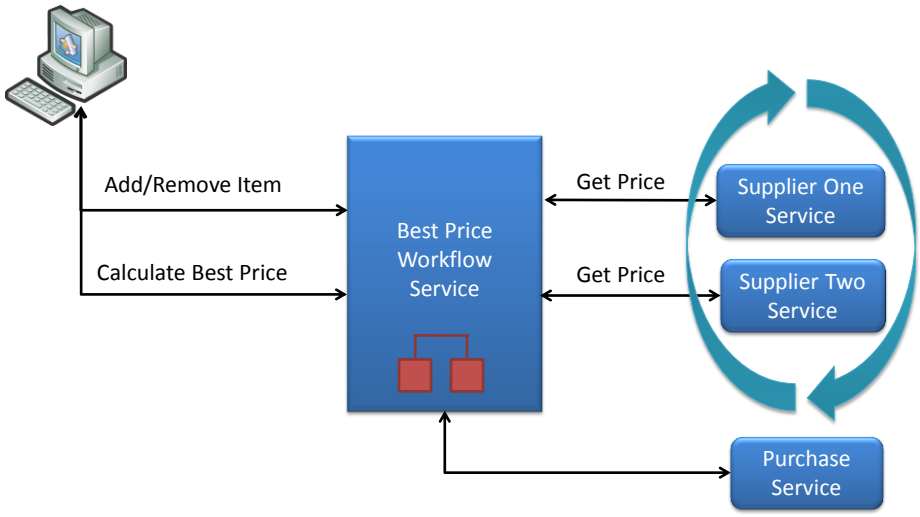
L'Environment di riferimento per WCF e WF 4

WF Service Components



Componenti di riferimento per un WF Service

WF Service: coordinazione di chiamate asincrone a servizi



WF Service Demo

DEMO

- Creazione e configurazione di un WF Service

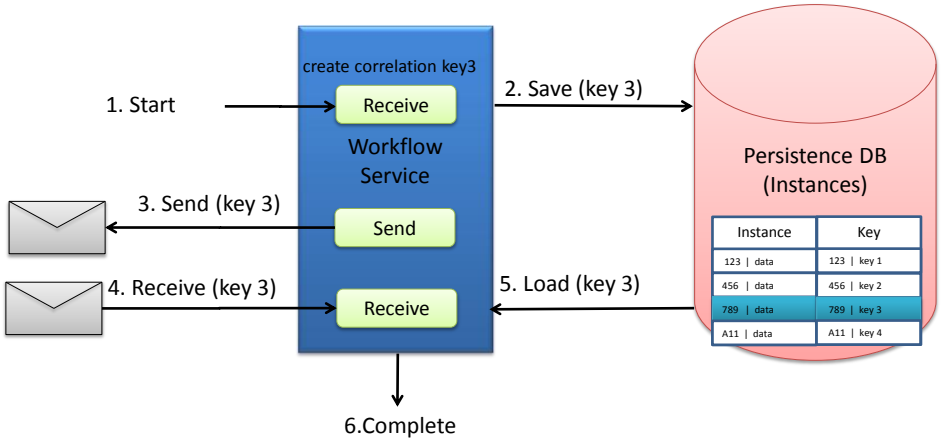
WF Service: CONTEXT e DATA Correlation

Le invocazioni multiple tramite l'Activity «Recive», richiedono un modo per riconoscere a quale istanza riferirsi.

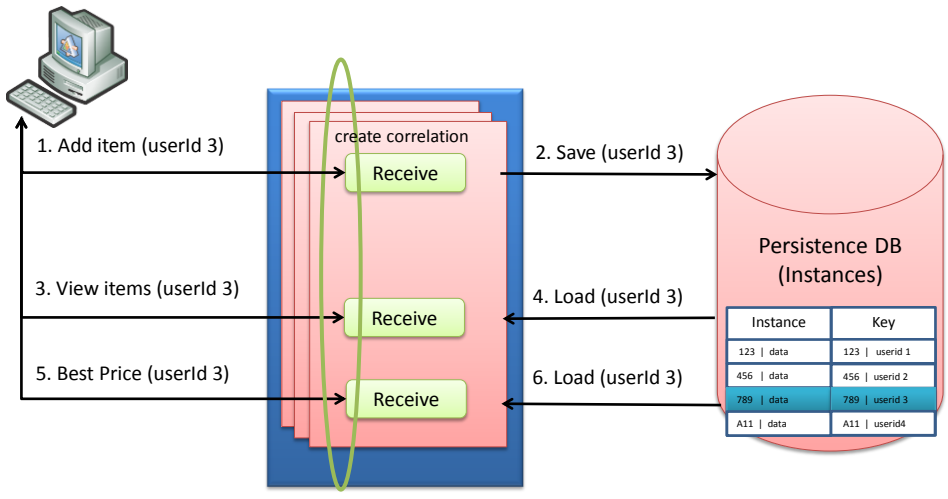
Per fare ciò esistono due possibilità:

- **Context Based Correlation**, in cui la chiave di correlazione è parte del binding. Questa modalità richiede lato client l'utilizzo del Binding WSHttpContextBinding.
- **Content Based Correlation** (aggiunto con il .**NET 4**), din cui la (le) chiave(i) di correalazione è passata come parte del messaggio. Adatta ad un uso più general purpose e cross-platfrom.

WF Service: CONTEXT BASED CORRELATION



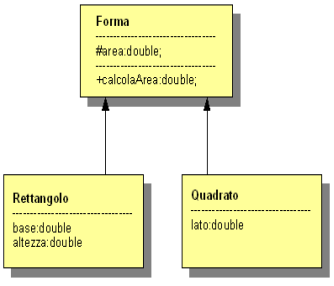
WF Service: CONTENT BASED CORRELATION



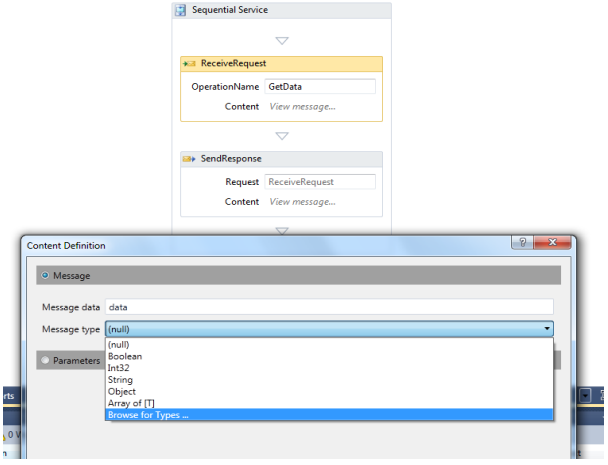
WF Service: CONTENT BASED CORRELATION le limitazioni del Designer

Il designer non è in grado di effettuare la correlazione sfruttando proprietà degli oggetti padre nel caso in cui il messaggio passato all'activity *Receive* sfrutti l'ereditarietà. Inoltre viene settato solo il namespace del figlio, che non permette di accedere ai parametri del padre se quest'ultimo ha un namespace diverso.

Prendiamo in considerazione la seguente gerarchia di oggetti (ovvero DataContract)



se all'oggetto Receive passiamo un messaggio (message) di tipo **Rettangolo,** non riusciremo, tramite il designer, a settare correttamente una correlazione basata sul parametro **Area** del padre.



Per usare le proprietà dell'oggetto padre bisogna passare alla modalità codice (View Code, F7), identificare il segmento xaml relativo alla Receive, e settare manualmente i parametri relativi all'**XPathMessageQuery**.

Il seguente esempio assume come namespace del padre: ***http://mydatacrontract*** e del figlio: ***http://mydatacontract/figure***

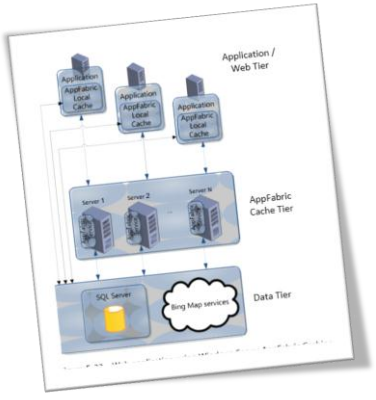
```
<QueryCorrelationInitializer CorrelationHandle="[correlator]">
    <XPathMessageQuery x:Key="key1">
     <XPathMessageQuery.Namespaces>
      <ssx:XPathMessageContextMarkup>
       <x:String x:Key="xg0">http://mydatacrontract</x:String>
       <x:String x:Key="xg1"> http://mydatacrontract/figure</x:String>
      </ssx:XPathMessageContextMarkup>

</XPathMessageQuery.Namespaces>sm:body()/xg1:Rettangolo/xg0:Area</XPathMessageQuery>
 </QueryCorrelationInitializer>
```

WF Service Demo

DEMO

- Content Correlation

Parte Quarta

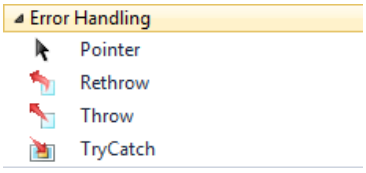


ASPETTI AVANZATI

Aspetti Avanzati:
ERROR HANDLING, EXCEPTION e FAULT

Built-in Activities: Error Handling

- **Rethrow**
- **Throw**
- **TryCatch**



Il ramo FINALLY dell'Activity Try/Catch non si comporta come l'omonima istruzione dei linguaggi: praticamente viene eseguita solo se viene eseguito il **blocco try** o uno dei **blocchi catch**.
Se si verifica un'eccezione non catturata, neanche il Finally verrà eseguito causando l'Abort del Flow. Nel caso di Fault da Servizio (Send/Recive Activity) l'attuale implementazione dell'Activity di Try/Catch è in grado di gestire solo la base class FaultExeception e non la generica FaultEception<MyFault>. Se si prova a catturare quest'ultima l'eccezione verrà propagata.

Suggerimento: se si è anche autori del servizio che può generare il Fault, utilizzare uno dei parametri standard di FaultException (ad esempio Action) per individuare l'eccezione specifica.

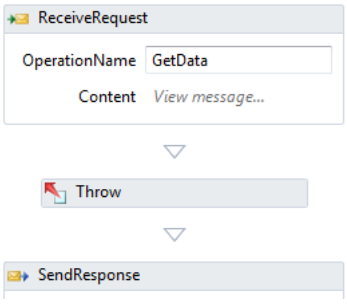
WF Service: Fatal and NonFatal Error

- Potendo realizzare dei Workflow utilizzabili in tutto e per tutto come Servizi, è normale pensare ad una gestione oculata dei Fault.

- In WFF possiamo suddividere le eccezioni (fault) in due gruppi:
  - **Fatal Error**, sollevate tramite l'Activities *Throw*
  - **NonFatal Error**, sollevate tramite una Replay specifica od *una Rethrow* all'interno di uno dei Catch dell'activity TryCatch

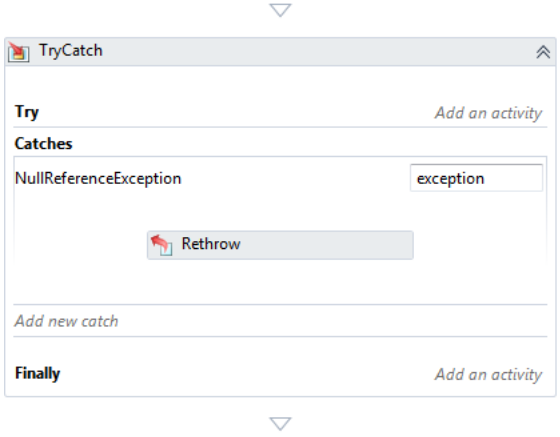
WF Service: Fatal Error

- Sollevo un Exception/Fault specifico abortendo di fatto l'esecuzione del WF



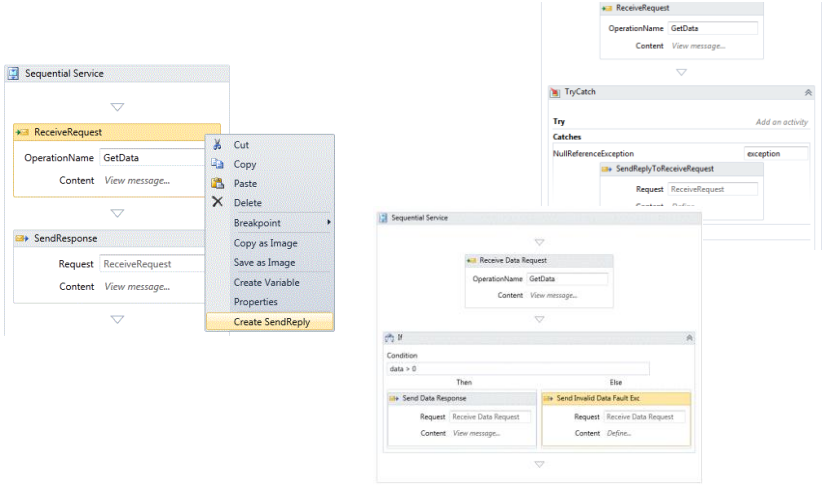
WF Service: FATAL ERROR

- Sollevo un Exception/Fault



WF Service: NON FATAL ERROR

- Propago un Fault Specifico come risposta all'invocazione



DotNet Campania team

Parte Quarta

Windows Server AppFabric

Aspetti Avanzati:

WINDOWS SERVER APPFABRIC
MONITORING,PERSISTENZA e CACHE

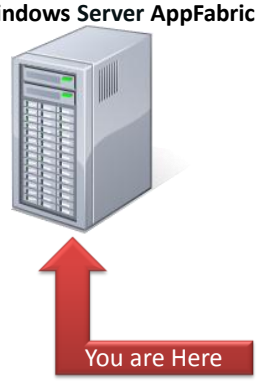
DotNet Campania team

Windows Server AppFabric

Windows Server **AppFabric** forinesce una serie di servizi per supportare i servizi WCF/WF e le Web App

- **Hosting, Monitoring e Controllo dei servizi WCF, in particolare per i WF Service;**
- **Ambiente unificato per le fasi di DEVELOPMENT e di Production**
- **Sevizi di Caching per le applicazioni ASP.NET e per la logica dei servizi**

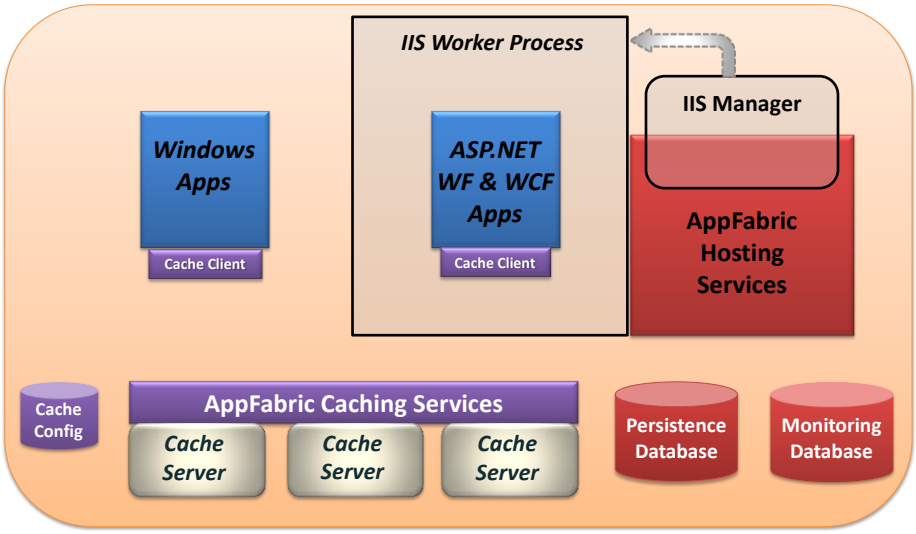
Windows Server AppFabric



Windows Workflow Foundation **Parte Quarta**

DotNet Campania team

Windows Server AppFabric: Architettura



DotNet Campania team

Parte Quarta

Windows Server AppFabric

MONITORING

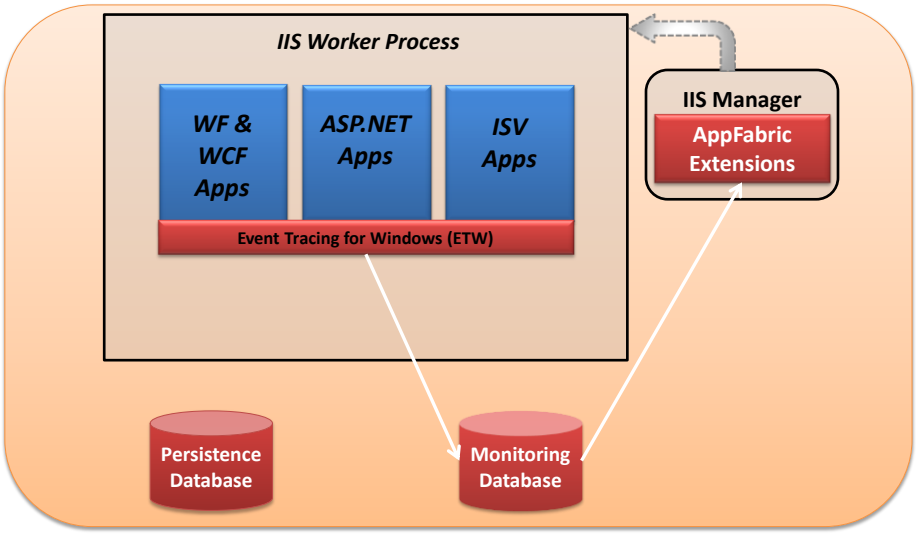
AppFabric Hosting Services: Monitoring

- Strumenti integrati con *Event Tracing for Windows (ETW),* per uniformare la gestione degli eventi (*warning*, *info*, *error*) dei servizi WF/WCF
  - WCF trace events
  - WF tracking events
- Performance migliorate per ridurre al minimo l'impatto sui servizi/applicazioni
- Granluarità delle informazioni

Scenario

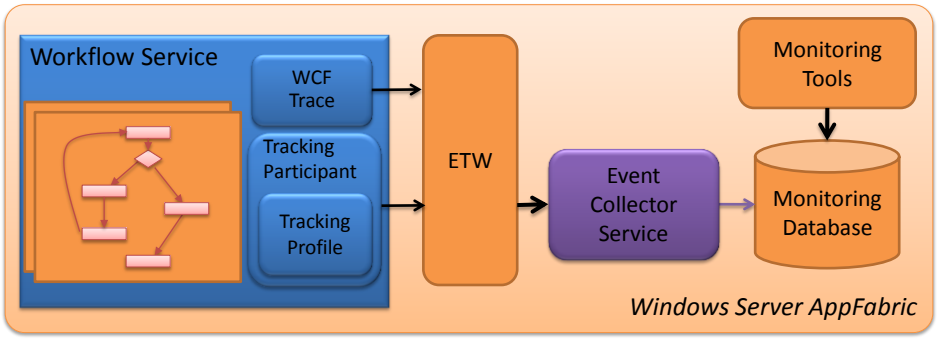
- Health monitoring – "Come sta funzionando la mia applicazione?"
- Troubleshooting – "Cos'è andato storto?"

AppFabric Hosting Services: Monitoring



DotNet Campania team

AppFabric Hosting Services: Monitoring, ECS



Event Collector Service (ECS)

- Si tratta di un Windows Service installato con *Windows Server AppFabric*
- Raccoglie eventi da WCF e WF scrivendoli nel Monitoring DB

DotNet Campania team

Parte Quarta

Windows Server AppFabric

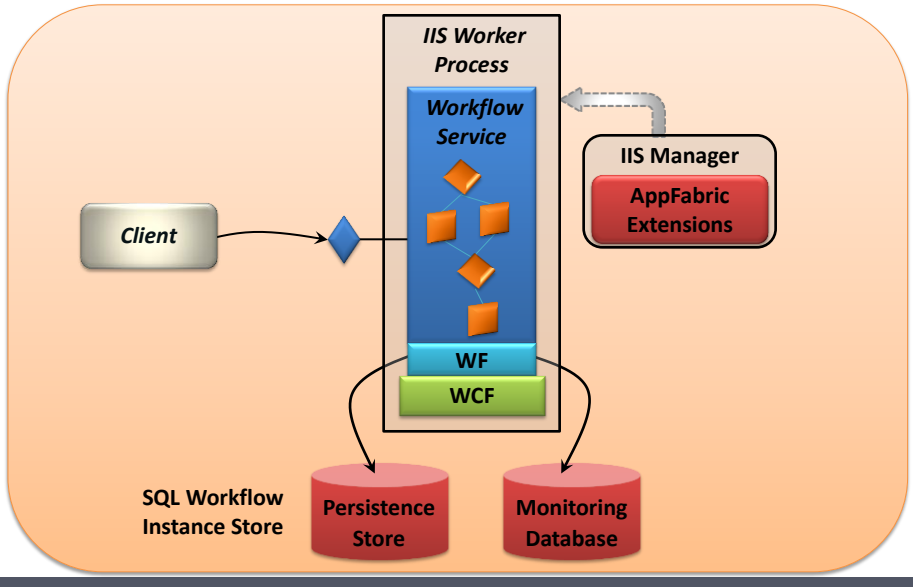
PERSISTENZA e VERSIONING

AppFabric Hosting Services: Persistenza

- Attraverso la «persistenza» è possibile sospendere l'esecuzione di un Workflow, salvarlo (da qualche parte) e riprenderne l'esecuzione successivamente

- Permette di garantire il recovery ed il resume di istanze di WorkflowScenarios
  - **Reliability**: Long running workflows
  - **Availability**: Recovery quando un'applicazione o una macchina va in crash
  - **Scalability**: L'istanza di un workflow viene deallocata dalla memoria e ricaricata solo quando necessario.

AppFabric Hosting Services: Persistenza



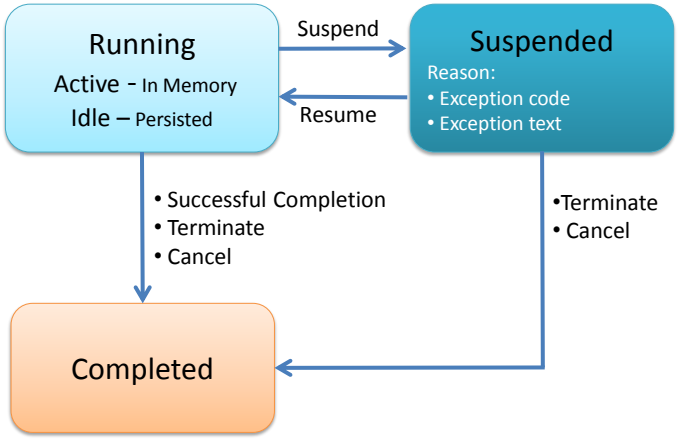
AppFabric Hosting Services: Persistenza

- Modalità di attuazione:
  - *Workflow (Persist Activity)*
  - *WorkflowApplication (Idle)*
  - *Host (wa.Persist())*

Di default il framework fornisce quanto necessario per effettuare la persistenza con SQLServer, unitamente agli script per creare le tabelle necessarie:

```
osql -E -S .\sqlexpress -Q "Create Database WorkflowInstanceStore"
osql -E -S .\sqlexpress -d WorkflowInstanceStore -i SqlWorkflowInstanceStoreSchema.sql
osql -E -S .\sqlexpress -d WorkflowInstanceStore -i SqlWorkflowInstanceStoreLogic.sql
```


Workflow: Stati in relazione alla Persistenza



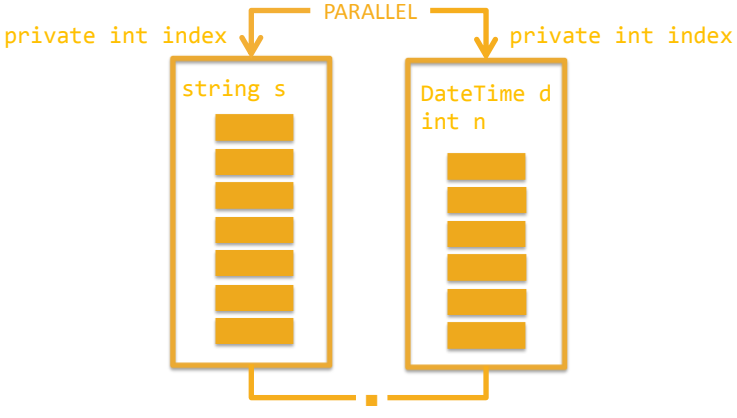
Operator Commands:

• Suspend, Resume, Terminate and Cancel

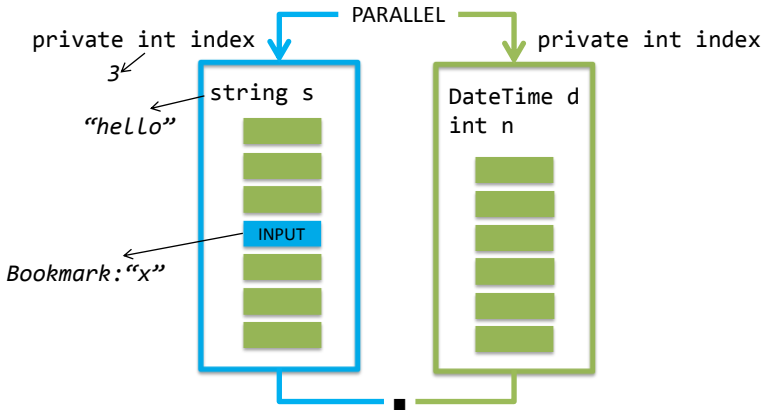
AppFabric Hosting Services: Cosa viene salvato con la persistenza?

- Un'istanza di wf serializzata contiene:
  - la lista dei work item (vuota se l'istanza è in idle, come nel caso dell'attesa di un input)
  - i Bookmark
  - i Dati (argomenti e variabili)
    - Il context di tutte le activity in esecuzione=> sospesione
  - Informazioni relative alle istanze delle Activity (callbacks, execution props)
  - Dati aggiuntivi
- La definizione del WF non viene salvata!
  - Milioni di istanze possono condividere la stessa definizione
  - E' compito dell'host gestire la definizione del WF persistito

AppFabric Hosting Services: WF Definiton



AppFabric Hosting Services: WF Serialized Data



AppFabric Hosting Services: Versioning!

Ecco uno dei problemi più spinosi di WWF: il **VERSIONING**!!!!

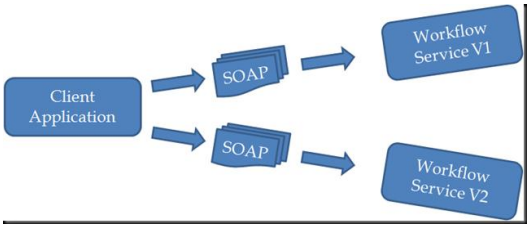
Perché spinoso:

- Non direttamente supportato dal framework
- Necessario effettuare dei workaround
- In caso di modifica della definizione di un WF tutte le istanze attive resteranno in un limbo perché non saranno riattivabili, se non previo ripristino della definizione originale.

AppFabric Hosting Services: Versioning, Rounting Workaround!

Quando l'SQL Workflow Instance Store effettua il salavataggio dell'istanza tiene traccia anche dell'indirizzo (address) WCF utilizzato per attivarlo.

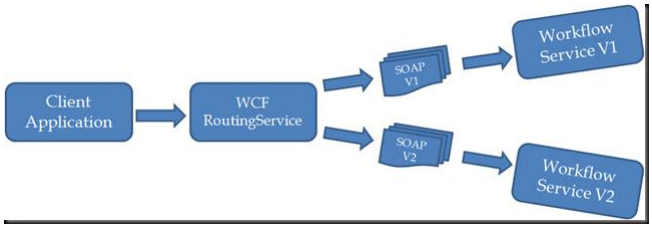
Il modo più banale per risolvere il problema del versioning sarebbe quello di cambiare indirizzo ad ogni nuova definizione di WF.



Questa soluzione in un contesto reale è però assolutamente inapplicabile perché costringere il client ad aggiornare ogni volta il proxy.

AppFabric Hosting Services: Versioning, Rounting Workaround!

Una soluzione elegante è quella di utilizzare il RoutingService di WCF4: in sostanza il client non interroga direttamente il servizio ma chiede di elaborare la propria richiesta ad un servizio intermedio di routing che conosce le varie versioni (indirizzi) della definizione dei WF ed è in grado di ridimenzionare la richiesta a quella corretta.



Come accade questa magia? Beh, come sempre, i modi sono diversi, ma il più semplice è quello di far ritornare in seguito alla prima invocazione il «version number» della definizione e riutilizzarlo per tutte le relative chiamate successive. Se non è presente un «version number», la richiesta viene indirizzata sempre all'ultima definizione, creando una nuova istanza.

AppFabric Hosting Services: Versioning, Rounting Workaround!

Il RoutingService è quindi la panacea a tutti i mali?

Sfortunatamente no: infatti se è presente un bug nella definizione legata a una serie considerevole di WF persistiti, non vi è alcun modo di porre rimedio, ovvero ricaricare le istanze in una versione rivista.

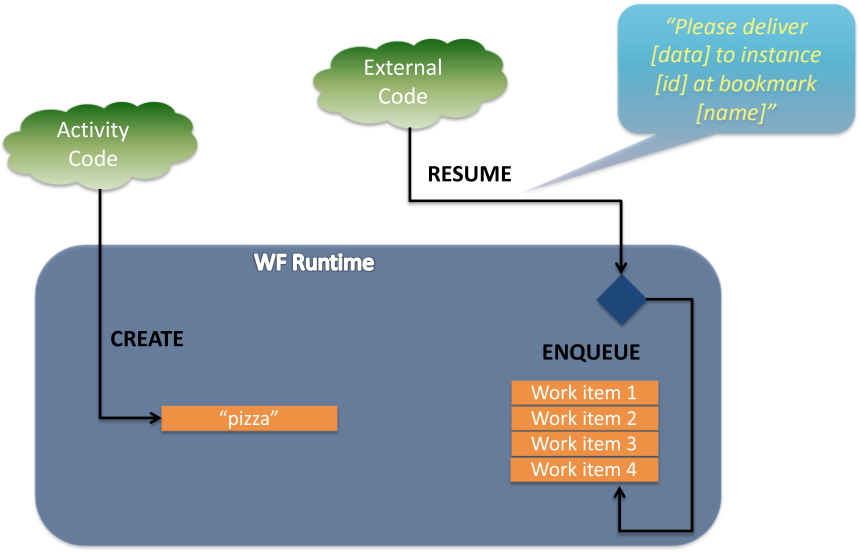
Per questo bisognerà almeno vNext, alias .NET 4.5, dove verrà introdotta la funzionalità denominata *Dynamic Updates* che dovrebbe permettere di identificare la definizione corretta.

In attesa che BigM risolva la questione, il consiglio è: definite con estrema precisione il flusso ed effettuate quanti più test possibili su di esso!

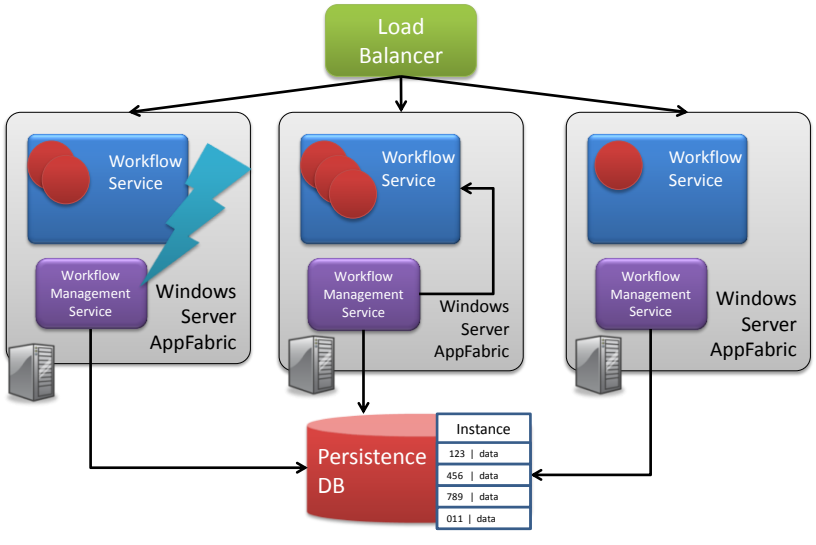
Bookmarks

- Cosa accade quando il vostro programma deve aspettare (anche molto) per ottenere dei dati di input?
  - … e cosa accade quando ci sono 1000 istanze in attesa?
- Per evitare un idle decisamente lungo, possiamo sfruttare i WF **bookmark**!
  - Rappresentano un punto ben identificato (nome) da cui riprendere l'esecuzione di un WF;
  - Il ripristino schedula un metodo di callback dell'Activity;
  - Il riavvio è gestino autonomamente dal runtime, evitando la necessità di matenere in memorie le istanze.
  - WCF Receive activity is built on top

Bookmark Resumption

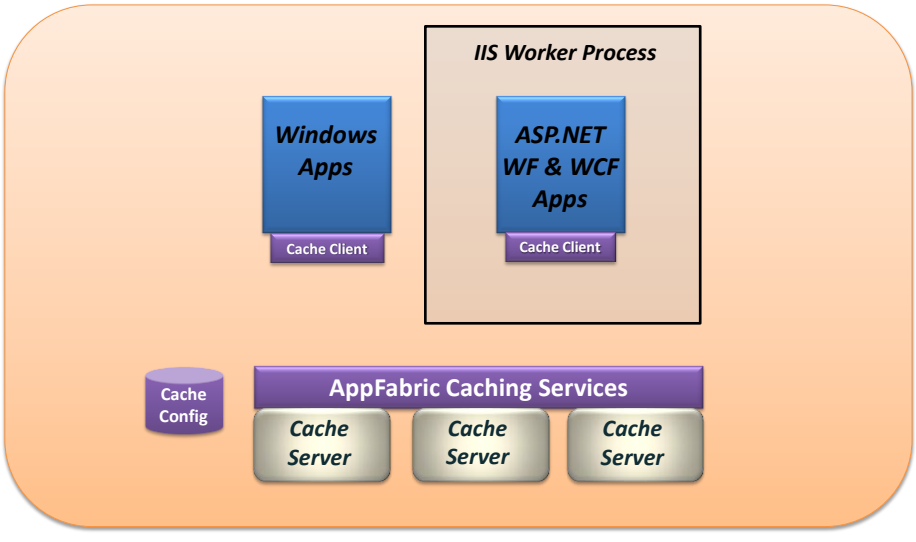


AppFabric Hosting Services: Persistenza e Load Balancer



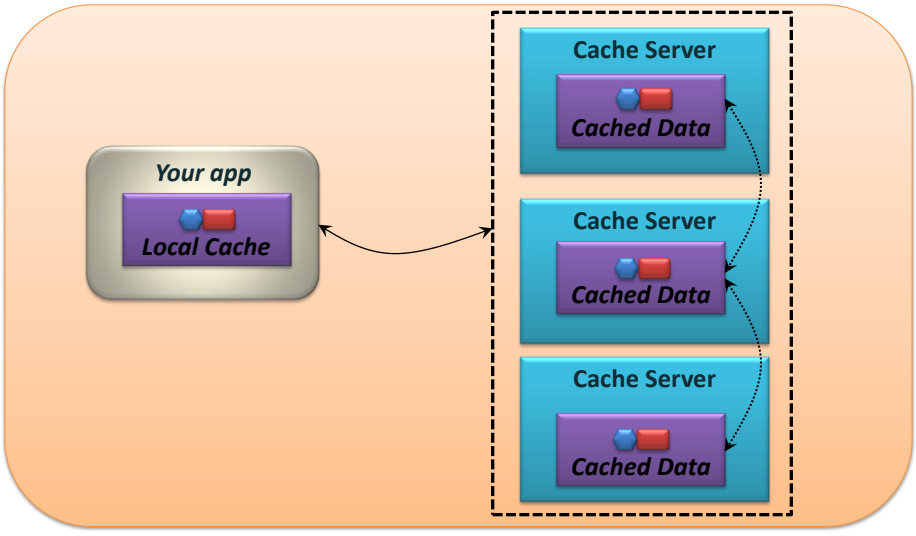
Windows Server AppFabric

CACHING



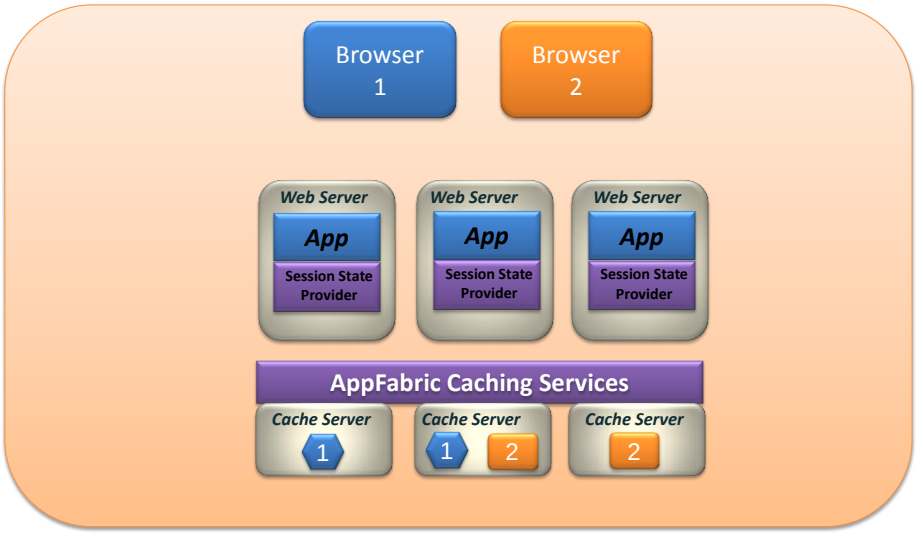
DotNet Campania team

High Availability



DotNet Campania team

Session State



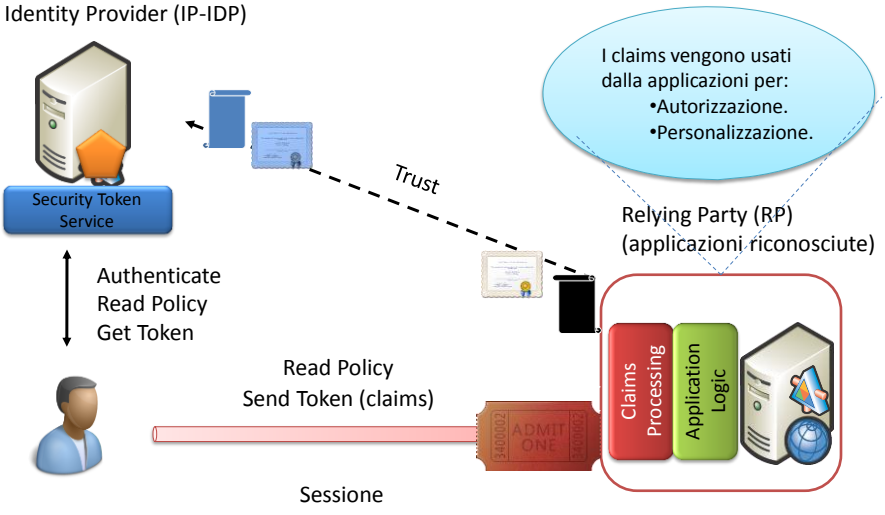
Microsoft .NET | Windows Identity Foundation

SECURITY:
introduzione a WIF – WINDOWS IDENTITY FOUNDATION

Claims-Based Security

- Normalmente una applicazione ottiene le informazione di identità di base
  - Es: username e password
- Successivamente l'applicazione deve fare un ulteriore query:
  - DB Locale o Remoto.
  - LDAP
- La claims-based security permette di estrapolare l'autenticazione dalle singole applicazioni
  - Prevede dei protocolli affinchè ogni applicazione possa richiere esattamente i claims che necessita
  - E' l'IDP che si preoccupa di ricavare gli attributi e creare i claims.
  - Le applicazioni non dovranno più accedere agli LDAP o a DB degli utenti.

Claims-Based Security



Claims-Based Security: Windows Identity Foundation

- E' un assembly .NET
- Permette alle applicazioni .NET di integrarsi con il modello claims-based.
- Completamente integrato con ASP.NET & WCF
- Unico programming model per on-premises & cloud
- Config driven
- Configurazione tramite Tools

Claims-Based Security: Security Token Services

Claims-Based Security: Sviluppo del servizio STS

- WIF fornisce dei "building blocks" per lo sviluppo di un proprio servizio STS
  - ADFS (Active Directory Federation Services) 2.0 è realizzato con WIF!
- Alcuni modelli di sviluppo per le diverse opzioni di hosting:
  - Active: WCF
  - Passive: ASP.NET
- Wizards e Templates per la creazione di un prototipo «scheletro» STS
  - Perfect for testing purposes

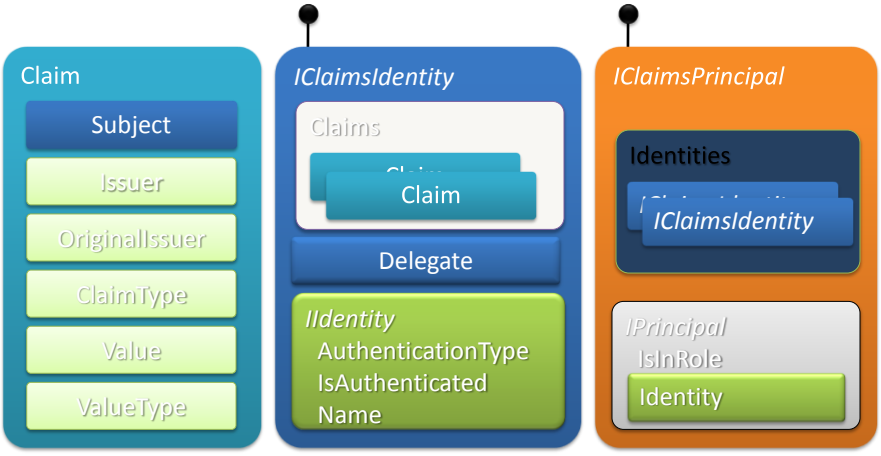
Attività principali nello sviluppo di un STS

- Decidere a chi consentire l'autenticazione (servizi, siti web, ecc.)
- Decidere il tipo di credenziali da accettare
- Fornire i certificati
- Realizzare la logica per recupere i claims

Claims Based Security: ws2007FederationHttpBinding

- L'utilizzo di un servizio STS è del tutto trasparente rispetto al servizio (WCF o WF Service) che si implementa.
- Infatti lo stack di attivazione può essere completamente configurato da web.config, sfruttando adeguatamente il binding ws2007FederationHttpBinding e configurando correttamente le Relying Party

Claims-Based Security: accesso ai Claim



DotNet Campania team

Claims-Based Security: accesso ai Claim

```
void Page_Load(object sender, EventArgs e)
{
  IClaimsPrincipal icp = (IClaimsPrincipal)
       Thread.CurrentPrincipal;

  IClaimsIdentity claimsIdentity =
       (IClaimsIdentity)icp.Identity;

  ageClaimValue = (from c in claimsIdentity.Claims
                   where c.ClaimType == "http://MyNS/AgeClaim"
                   select c.Value ).Single();
 }
```


DotNet Campania team

Parte Quinta

Security
WFF e WIF

1. **Claims-Enabled Workflow Service**

   Essendo un Workflow Services a tutti gli effetti un servizio WCF, WIF può essere integrato in modo naturale intervenendo sulla configurazione secondo quanto riportato su MSDN. In tal modo il WF Service diventa claims-enable ed è possibile sfruttare le estensioni WIF (code-based) per manipolare i claims (ClaimsAuthenticationManager) e per gestire l'autorizzazione claims-based (ClaimsAuthorizationManager).

2. **Workflow Services calling other Claims-Enabled Services**

   In un servizio WFC "standard" è possibile continuare la catena di invocazione claim-based sfruttrando il binding wsFederationHttpBinding o le API fornite da WIF API (WSTrustChannelFactory).

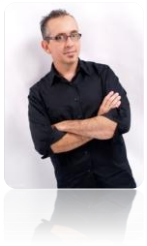
   In linea generale l'utilizzo del *wsFederationHttpBinding* è similare anche nel caso dei WF Service, anche se risulta particolarmente costoso considerando l'aspetto delle performance. Inoltre non è possibile utilizzarlo in tutti gli scenari, in particolare dove è necessario un controllo granulare sul TOKEN stesso.

3. **WIF in middle-tier Workflow Services**

   WIF consente la claims-based-delegation implementando le funzionalità di ActAs/OnBehalfOf del protocollo WS-Trust. Nei servizi «standard» è possibile utilizzare queste modalità, nei layer intermedi, ottenendo il Token SAML ed impersonando il chiamante per fare una chiamata in sua vece. Claims-based delegation non è agevolmente possibile nei WF Service tramite l'utilizzo di wsFederationHttpBinding.

Per consentire una gestione flessibile del Token SAML e degli altri aspetti legati a WIF, su CodePlex è possibile reperire il WF Security Pack (CTP 1).

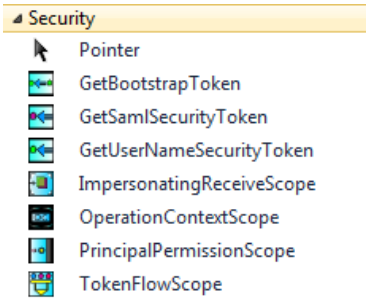
Si tratta di una serie di Activities aggiuntive che permettono un'interazione trasparente con WIF



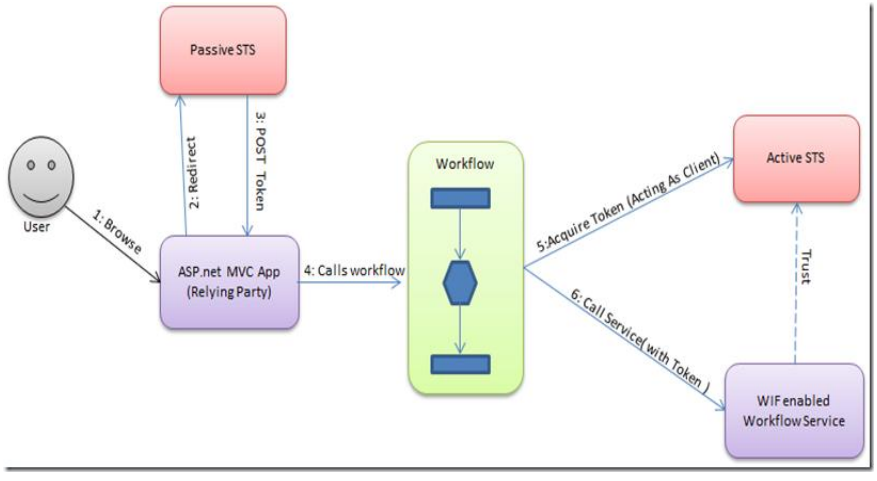
Il progetto è curato da Ron Jacobs, Sr. Program Manager di WF, WCF, appFabric e del .NET Framework stesso

WFF e WIF: WF Security Pack

- **GetBootstrapToken**
  Permette di ricavare il BootStrapToken, ovvero il Token originale emesso dall'STS e necessario per la claim-based-delegation
- **GetSamlSecurityToken**
  Ottiene il Security Token
- **GetUserNameSecurityToken**
  Crea uno "UsernameSecurityToken" basato sulla coppia Username/Password
- **ImpersonatingReceiveScope**
  Ricerca una WindowsIdentity all'interno del SecurityContext. Se trovato, tutte le Activities figlie verranno eseguite all'interno del relativo scope.
- **OperationContextScope**
  Permette di definire lo scope all'interno del quale verranno eseguite le Activity figlie
- **PrincipalPermessionScope**
  Consente di effettuare dei check sui claim prima di «autorizzare» l'esecuzione delle Activities figlie
- **TokenFlowScope**
  Trasferisce il Token (ottenuto tramite GetSamlSecurityToken) a tutte le Activity Send in esso contenute



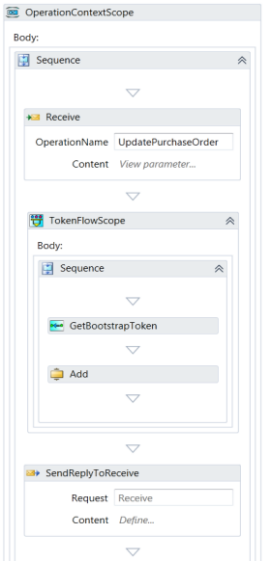
WFF e WIF: Scenario, pt1



NB: per accedere direttamente al BootStrapToke(RAW) come parte di *IClaimsIdentity*, è necessario configurare il parametro **SaveBootstrapToken** a TRUE.

WFF e WIF: Scenario, pt2

1. L'utente arriva ad una web application (relying party) che richiede autenticazione
2. WIF ridireziona l'utente verso l'STS in modalità passiva (tipicamente viene presentata una web-form di loing)
3. Dopo l'autenticazione, la richiesta viene re-diretta alla pagina ASP.net originale dove WIF verifica la validità del Token e l'utente accedere all'area riservata.
4. L'utente seleziona un pulsante (o link, o similare) che invoca il Workflow Service.
5. In questo caso usiamo le activity del SecurityPack (in modo innestato) per propagare il BootStrapToken al chiamante.
6. WF effettua le operazioni richieste propagando il context di security all'Activity Add
7. Viene ritornato il risultato



Parte Finale

Conclusioni e Prossimi Sviluppi

Evoluzione della specie

Come è noto, Microsoft è al lavoro su vNext (.NET 4.5), che nel caso dei Workflow introduce:

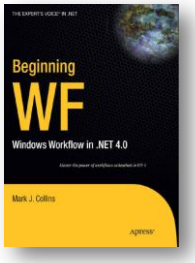
- *C# expressions*
- *Annotazioni (tipo post it sul wf);*
- *Auto-connect (drag&Drop)*
- *State Machine (reintroduzione ufficiale);*
- *Multi-assign*
- *SQL/State machine activities derivate direttamente da Codeplex*
- *Http Activities (POST/GET/…)*
- *Dynamic Update (per il versioning)*
- *…. altro?*

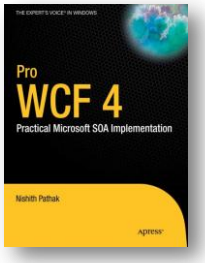
Sommario

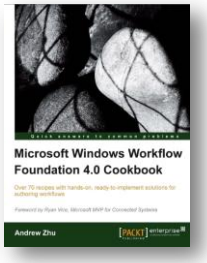
- Workflow è pensato principalmente per i processi/servizi di business
- I Workflow Service gestiscono la complessità di coordinamento tra più processi/servizi
  - Composizione di servizi
  - Correlazioni di messaggi
- Windows Server AppFabric supporta WCF e i Workflow Services
  - Funzionalità di Runtime
  - Monitoring, Persistenza, Hosting e Caching
  - Tool e script per la gestione ed il monitoring

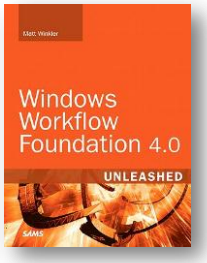
DotNet Campania team

Libreria









Microsoft

Escluso MSDN, la casa di Redmond latita un po' su testi e materiale ufficiale inerente WWF

DotNet Campania team

Blog e Riferimenti

Zulfiqar's weblog - WCF/WF/AppFabric & random .Net stuff

http://zamd.net/

Ron Jacobs - Windows Workflow Foundation

http://blogs.msdn.com/b/rjacobs/

The Problem Solver

http://msmvps.com/blogs/theproblemsolver/archive/tags/Workflow/default.aspx

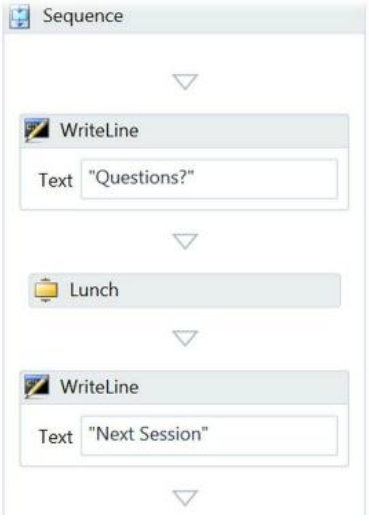
Riferimenti

http://dotnetside.org/blogs/articoli/pages/introduzione-a-windows-workflow-foundation-4.aspx
http://www.slideshare.net/dannicola/wf-40-overview
http://www.slideshare.net/markginnebaugh/microsoft-windows-server-appfabric

http://blogs.msdn.com/b/rjacobs/
http://msdn.microsoft.com/it-it/magazine/gg598919.aspx
http://xhinker.com/post/WF4Authoring-Workflows-Using-Imperative-Code.aspx

DotNet Campania team

Domande?



DotNet Campania team

Chi non riesce più a provare stupore e meraviglia è già come morto e i suoi occhi sono incapaci di vedere.

..:: Albert Einstein ::..