Programmazione Orientata agli Oggetti in Linguaggio Java

Strumenti di Sviluppo: C#

versione 1.0

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- IDE per .NET
- Nant
  - ⇨Preliminari
  - ⇨Differenze con Ant
- MSBuild
- Altri Strumenti
- File di Configurazione dell'Assembly

IDE per .NET

- L'IDE per eccellenza per .NET
  - ⇨Visual Studio.NET
  - ⇨discende da Visual Studio 6.0
  - ⇨varie versioni (VS.NET 2002, 2003, 2005)
  - ⇨prodotto commerciale distribuito in varie versioni
  - ⇨si tratta di un ambiente multilinguaggio che consente di sviluppare applicazioni con tutti i linguaggi principali della piattaforma .NET

IDE per .NET

- Il programma Microsoft Express
  - ⇨la Microsoft renderà disponibili gratuitamente una versione express dei suoi strumenti
  - ⇨le versioni express di Visual Studio sono versioni ridotte e monolinguaggio
  - ⇨http://lab.msdn.microsoft.com/express/
- Per il linguaggio C#
  - ⇨Visual C# 2005 Express Edition

IDE per .NET

- I progetti di Visual Studio
  - ⇨sono definiti “soluzioni”
  - ⇨contengono un progetto principale
  - ⇨e possono contenere un certo numero di sottoprogetti (librerie)
  - ⇨quando viene costruita la soluzione vengono costruiti tutti i progetti che la compongono

IDE per .NET

- Definizione del progetto
  - ⇨vari modelli predefiniti
  - ⇨scelta del nome
  - ⇨scelta della cartella del disco corrispondente
- Definizione dei riferimenti
  - ⇨insieme di assembly esistenti che vengono utilizzati – oltre agli assembly dei sottoprogetti – durante la compilazione e l’esecuzione

IDE per .NET

>> Visual Studio 2005

- Build di debug e build di release
  - una significativa differenza rispetto a Java
  - gli assembly prodotti sono significativamente diversi
  - oltre alle opzioni di debug e di ottimizzaz., tipicamente l'assembly di debug contiene anche i test, quello di release no
  - Visual Studio consente di definire le due diverse configurazioni e usa cartelle diverse

IDE per .NET

- Un IDE open source per .NET
  - SharpDevelop
  - distribuito su http://www.icsharpcode.net
  - riproduce le funzionalità essenziali di Visual Studio
- Installazione di SharpDevelop
  - distribuito sotto forma di installabile (.msi)
  - all'installazione è necessario creare una base di dati per il completamento del codice

IDE per .NET

>> SharpDevelop

- Progetti di SharpDevelop
  - ⇨sono detti “combinazioni”
  - ⇨analoghi alle soluzioni di Visual Studio
- Configurazione del progetto
  - ⇨tutti i file del progetto vengono registrati nei metadati
  - ⇨per cui se i file vengono aggiunti o eliminati al di fuori dell’IDE non risultano subito visibili
  - ⇨è utile specificare l’opzione per cui i nuovi file devono essere aggiunti automaticamente al progetto all’apertura

Nant

- Nant: “Not Ant"
  - ⇨strumento open source per .NET scaricabile dal sito http://nant.sourceforge.net
- Storia di Nant
  - ⇨deriva da un progetto chiamato XBuild, poi soppresso, che era ispirato ad Ant
  - ⇨sviluppato replicando abbastanza fedelmente le caratteristiche di Ant
  - ⇨attualmente alla versione 0.85

Nant

- Installazione di Nant
  - ⇨viene distribuito sotto forma di .zip
  - ⇨basta decomprimere lo .zip in una cartella
  - ⇨e aggiungere %NANT_HOME%\bin al PATH
- I task di Nant
  - ⇨circa 70
  - ⇨non esiste la distinzione tra task principali e task opzionali

Nant

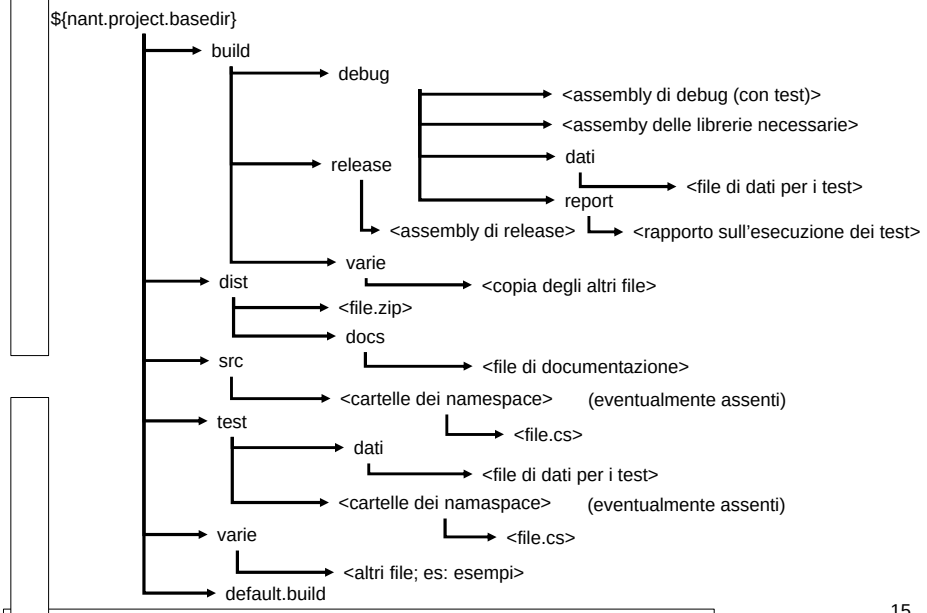
- Librerie necessarie
  - ⇨per l'esecuzione dei test di regressione è necessario NUnit
  - ⇨per la generazione della documentazione è necessario NDoc
  - ⇨per la creazione di file zip è necessaria la libreria di SharpZipLib
  - ⇨tutte e tre le librerie vengono distribuite con nant per cui non è necessario installare altro

Nant

- Il progetto nantcontrib
  - fornisce 88 task aggiuntivi rispetto a quelli fondamentali
  - attualmente alla versione 0.85
  - i task vengono distribuiti sotto forma di file .zip
  - è necessario decomprimere lo .zip e poi copiare il contenuto della cartella bin in %NANT_HOME%\bin

Preliminari

- Struttura della cartella di progetto
  - simile a quella standard per Ant
- In particolare
  - il codice sorgente è contenuto in una cartella src, con sottocartelle per i namespace
  - i test sono tenuti nella cartella test
  - il codice oggetto è tenuto in build, con due sottocartelle separate (debug e release)
  - il codice distribuibile è tenuto in dist



Preliminari

>> utilita\default-semplice.build

- I file di build di Nant
  - ⇨sono file XML
  - ⇨tipicamente con estensione .build
  - ⇨contengono ciascuno un progetto
  - ⇨che è fatto di uno o più target
  - ⇨che contengono al loro interno uno o più task
- La sintassi di Nant
  - ⇨molto simile a quella di Ant

Preliminari

- Alcune particolarità
  - ⇨alcuni task falliscono in condizioni in cui i corrispondenti task di Ant non fallirebbero
  - ⇨es: delete, mkdir ecc.
  - ⇨in questi casi è necessario utilizzare l'attributo failonerror per evitare che il processo di build fallisca

Preliminari

>> ant su utilita

- Esecuzione di Nant
  - ⇨comando nant dal prompt
- Varianti
  - ⇨nant esegue il target di default del file default.build oppure cerca il primo file .build
  - ⇨nant <target>
  - ⇨nant –f:<nomeFile>.build (o –buildfile:, /f:)
  - ⇨nant –f:<nomeFile>.build <target>
  - ⇨-help, -verbose, -debug, -quiet, -projecthelp

Preliminari

>> SharpDevelop

- Nant e VisualStudio
  - ⇨VisualStudio 2005 ha il proprio sistema di build (MSBuild) (>>)
  - ⇨esiste però un plugin per consentire di utilizzare Nant
- Nant e SharpDevelop
  - ⇨non esiste un plugin specifico
  - ⇨è però possibile eseguire Nant come strumento esterno configurando il comando

Differenze Rispetto ad Ant

>> Nant docs: Proprietà predefinite

- Proprietà
  - ⇨le proprietà di Nant non sono necessariamente immutabili
  - ⇨per renderle immutabili è necessario specificare l'attributo readonly
  - ⇨sono sempre immutabili le proprietà passate attraverso la linea di comando
  - ⇨argomento –D:<nome>=<valore>
  - ⇨esistono molte proprietà predefinite

Differenze Rispetto ad Ant

- Formato per le proprietà
  - ⇨Nant (come .NET) è basato su XML e non su file .properties
  - ⇨per prelevare valori di proprietà da XML bisogna utilizzare il task xmlpeek, che consente di specificare un xpath per prelevare un valore da un file XML
  - ⇨per scrivere valori di proprietà è necessario utilizzare il task xmlpoke

Differenze Rispetto ad Ant

>> Nant docs: Espressioni e Funzioni

- Infine
  - ⇨Nant non rispetta la filosofia rigorosamente dichiarativa di Ant
- Costruzione “procedurale” in Nant
  - ⇨il linguaggio fornisce un supporto decisamente maggiore rispetto ad Ant
  - ⇨possibilità di scrivere condizioni complesse
  - ⇨possibilità di scrivere funzioni complesse
  - ⇨tag if, ifnot, foreach

Differenze Rispetto ad Ant

>> questionari\default-semplice.build

- Un esempio
  - ⇨build di debug e build di release
- Nel caso di .NET
  - ⇨bisogna costruire due assembly completamente diversi
  - ⇨è necessario eseguire task diversi
  - ⇨la soluzione: utilizzare il task call per "biforcare il flusso di esecuzione"

Differenze Rispetto ad Ant

- Il task csc
  - ⇨consente di eseguire il compilatore di C#

```
<target name="compile-release" depends="prepare, prepare-references-release">
    <csc target="exe" output="${build.release.dir}\${assembly.name}"
         debug="false“ verbose="false">
      <sources basedir="${src.dir}">
        <include name="**/*.cs" />
      </sources>
      <references basedir="${mylib.dir}">
        <include name="Unibas.Utilita.dll" />
      </references>
    </csc>
</target>
```

Differenze Rispetto ad Ant

- Attenzione ai riferimenti
  - ⇨il task csc consente di specificare solo i riferimenti per il compilatore
  - ⇨è poi necessario predisporre anche i riferimenti per il caricatore

```
<target name="prepare-references-release" depends="prepare">
    <copy todir="${build.release.dir}">
      <fileset basedir="${mylib.dir}">
        <include name="Unibas.Utilita.dll" />
      </fileset>
    </copy>
 </target>
```


Differenze Rispetto ad Ant

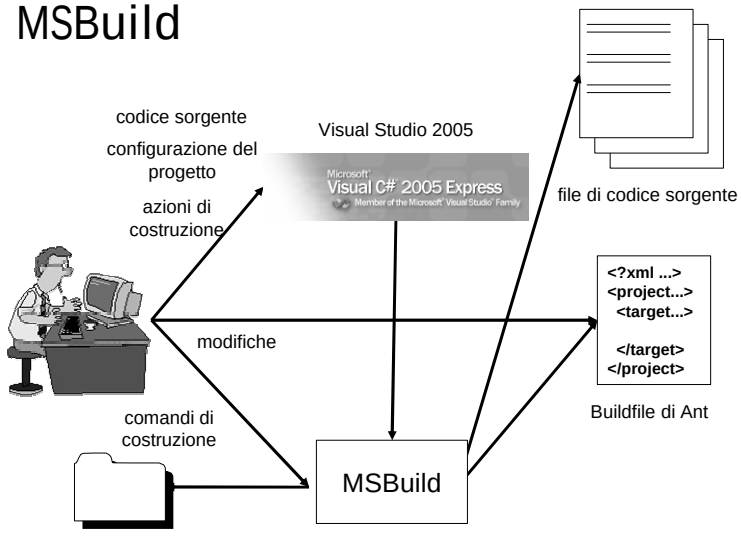
>> console-template.build
>> questionari\default.build

- Un limite di Nant
  - ⇨non esiste il task import
- Il task include
  - ⇨sostituisce parzialmente import, consentendo di importare i target di un buildfile in un altro
  - ⇨ma non consente sovrascritture
  - ⇨quindi è inutile nei casi in cui non tutti i target del template sono direttamente riutilizzabili

MSBuild

- Il nuovo Visual Studio
  - ⇨Visual Studio 2005, nome in codice Whidbey
  - ⇨integrerà un sistema di costruzione secondo l'impostazione di NetBeans 4.0
- MSBuild
  - ⇨sistema di costruzione derivato da Nant
  - ⇨verrà distribuito con tutte le nuove versioni di Windows (a partire da "Longhorn")

MSBuild



MSBuild

- Un esempio di buildfile

```
<Project MSBuildVersion="1.0" DefaultTargets = "Compile">
  <Property appname = "HelloWorldCS"/>
  <Item Type = "CSFile" Include = "consolehwcs1.cs"/>
  <Target Name = "Compile">
      <Task Name = "CSC" Sources = "@(CSFile)">
          <OutputItem TaskParameter = "OutputAssembly"
                      Type = "EXEFile" Include = "$(appname).exe"/>
      </Task>
      <Message Text="The output file is @(EXEFile)"/>
  </Target>
</Project>
```


Altri Strumenti

- Anche per .NET
  - ⇨esistono molti altri strumenti disponibili
- Esempi
  - ⇨Convenzioni di stile: FxCop http://www.gotdotnet.com/team/fxcop/
  - ⇨Profiler: NProf http://nprof.sourceforge.net
  - ⇨Analisi di Copertura: Coverageeye.net www.gotdotnet.com

File di Configurazione dell'Assembly

- Il task NUnit
  - ⇨nella versione 0.85 utilizza per l'esecuzione dei test NUnit v. 2.2 (nunit.framework.dll)
- Ma...
  - ⇨i test potrebbero essere stati compilati utilizzando una versione diversa (es: precedente)
  - ⇨in questo caso si crea un problema

File di Configurazione dell'Assembly

- Infatti
  - ⇨il collegatore non troverebbe corrispondenza tra la versione utilizzata dal compilatore e la versione utilizzata per l'esecuzione
- In questo caso
  - ⇨è necessario effettuare una redirezione da una versione all'altra di NUnit
  - ⇨attraverso il file di configurazione dell'assembly su cui eseguire i test

File di Configurazione dell'Assembly

- File di configurazione dell'assembly
  - ⇨file XML attraverso cui è possibile specificare una serie di parametri di configurazione per l'assembly
  - ⇨deve chiamarsi come l'assembly, con in più l'estensione .config
  - ⇨deve stare nella stessa cartella dell'assembly
  - ⇨es: questionari.exe.config, Unibas.Utilita.dll.config
  - ⇨viene caricato automaticamente dalla m.v.

File di Configurazione dell'Assembly

- Tipologie di informazioni
  - ⇨dati di configurazione a tempo di esecuzione
  - ⇨versioni del framework .NET supportate
  - ⇨valori acquisibili all'interno del codice per la configurazione (es: logging)
  - ⇨redirezioni tra versioni diverse dello stesso assembly utilizzato

>> nunit.framework.dll.config
>> nantconsole.exe.config

File di Configurazione dell'Assembly

- Nel caso di NAnt
  - ⇨per cambiare versione del framework NUnit è necessario creare un frammento di file di configurazione che produca la redirezione es: test.config
  - ⇨poi eseguire il task nunit specificando con l'attributo appconfig di utilizzare il file creato come file di configurazione per l'assembly

File di Configurazione dell'Assembly

```
<!-- test.config -->
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
         <dependentAssembly>
            <assemblyIdentity name="nunit.framework"
                              publicKeyToken="96d09a1eb7f44a77"
                              culture="Neutral" />
            <bindingRedirect oldVersion="2.0.6.0" newVersion="2.2.0.0" />
            <bindingRedirect oldVersion="2.1.4.0" newVersion="2.2.0.0" />
         </dependentAssembly>
      </assemblyBinding>
   </runtime>
</configuration>

<nunit2>
   <formatter type="Plain" />
   <test assemblyname="questionari.dll" appconfig="test.config" />
</nunit2>
```

Riassumendo

- IDE per .NET
- Nant
  - ⇨Preliminari
  - ⇨Differenze con Ant
- MSBuild
- Altri Strumenti
- File di Configurazione dell'Assembly

Termini della Licenza