Programmazione Orientata agli Oggetti in Linguaggio Java

Sintassi e Semantica:

C#

Parte b

versione 2.3

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

Sommario

- Il Compilatore
  - ⇨Il Concetto di Assembly
  - ⇨Compilazione e Collegamento
- Modificatori di Visibilità
- Commenti di Documentazione
  - ⇨NDoc
- Convenzioni di Stile di C#

Il Compilatore

- Il Compilatore di C#
  - csc.exe
- Attenzione
  - come detto, in C# non c'è nessuna relazione tra il contenuto del codice sorgente e la struttura dei file su disco
  - e il processo di compilazione, di collegamento e di esecuzione è totalmente diverso

Il Concetto di Assembly

- Il codice oggetto di .NET
  - viene confezionato sotto forma di assembly
- Assembly
  - file contenente il codice intermedio .NET di una o più classi
  - con le relative metainformazioni che le descrivono ("manifesto")
  - analogo di un .jar di Java

Il Concetto di Assembly

- Tipologie principali di assembly
  - ⇨exe: eseguibile per la console, estensione predefinita .exe
  - ⇨library: libreria dinamica, estensione .dll
- Inoltre
  - ⇨winexe: applicazione grafica
  - ⇨module: crea solo il codice oggetto senza le metainformazioni (in questo modo l'assembly può essere aggiunto ad altri assembly)

Il Concetto di Assembly

- Per compilare un file di codice C#
  - ⇨bisogna specificare l'opzione /target
- Esempi
  - ⇨csc /target:exe Prova.cs
    genera Prova.exe, eseguibile
  - ⇨csc /target:library Prova.cs
    genera Prova.dll, libreria collegabile dinam.
  - ⇨il valore di default è /target:exe (o /t:exe)

Compilazione e Collegamento

- Per creare un'applicazione completa
  - ⇨due opzioni
- Collegamento statico
  - ⇨compilo assieme tutte le classi dell'applicazione in un unico assembly
- Collegamento dinamico
  - ⇨compilo solo alcune delle classi nell'assembly e indico riferimenti ad assembly esterni

Compilazione e Collegamento

Attenzione: le librerie di .NET sono comunque collegate dinamicamente

- Collegamento statico
  - ⇨esempio: circonferenze
  - ⇨tre classi utilizzate: Circonferenza.cs, Principale.cs, Console.cs
  - ⇨oltre alle librerie di sistema
- Comando
  - ⇨csc /target:exe /out:Circonferenze.exe Principale.cs Circonferenza.cs ..\Utilita\Console.cs

Compilazione e Collegamento

- In questo modo, però...
  - ⇨si perdono tutti i vantaggi del collegamento dinamico
- Esempio
  - ⇨non è possibile sostituire la classe Console.cs con una versione successiva che fornisca gli stessi metodi in forma migliorata
- Di conseguenza
  - ⇨utilizzeremo il collegamento dinamico

Compilazione e Collegamento

- Collegamento dinamico
  - ⇨in questo caso posso compilare solo alcune delle classi in un assembly eseguibile
  - ⇨es: Principale.cs e Circonferenza.cs in .exe
  - ⇨ma devo risolvere due problemi
- Problema n. 1
  - ⇨specificare al compilatore dove sono gli altri componenti per effettuare le verifiche a tempo di compilazione (Unibas.Utilita.dll)

Compilazione e Collegamento

- Problema n.2
  - ⇨fare in modo che i componenti da collegare dinamicamente siano effettivamente reperibili a tempo di esecuzione (Unibas.Utilita.dll)
- Attenzione
  - ⇨questi stessi problemi in Java sono risolti attraverso la gestione del CLASSPATH
  - ⇨sia per il compilatore (javac), sia per la macchina virtuale (java)

Compilazione e Collegamento

- Problema n.1: Soluzione
  - ⇨il compilatore di C# consente di specificare a tempo di compilazione la posizione degli assembly esterni (namespace) utilizzati nel codice
- Opzione /reference:<elencoAssembly>
  - ⇨esempio: circonferenze
  - ⇨csc /target:exe /out:Circonferenze.exe /reference:Unibas.Utilita.dll *.cs

Compilazione e Collegamento

- Attenzione
  - ⇨l'opzione /reference NON dice di includere il codice di Unibas.Utilita.dll nell'assembly Circonferenze.exe
  - ⇨indica solo al compilatore dove andare a trovare l'assembly per effettuare le verifiche sul codice di Principale.cs
  - ⇨ma i due assembly restano separati tra di loro e dovranno essere collegati dinamicam.

Compilazione e Collegamento

- Problema n.2: Soluzione
  - ⇨a tempo di esecuzione, gli assembly esterni da collegare devono essere contenuti nella stessa cartella dell'assembly eseguibile
  - ⇨ovvero: Unibas.Utilita.dll deve essere nella stessa cartella di Circonferenze.exe
- Nota
  - ⇨la gestione degli assembly di .NET è decisamente più sofisticata (>>)

Compilazione e Collegamento

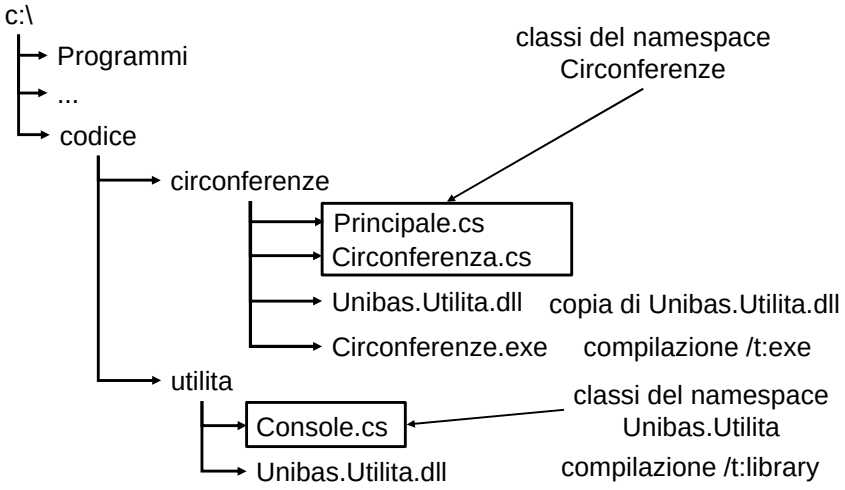
- Le librerie del .NET framework
  - sono contenute nell'assembly mscorlib.dll
  - in c:\windows\Microsoft.NET\framework\v.1.1.4322
  - sono automaticamente visibili al compilatore e alla macchina virtuale e non devono essere specificate
- Nota
  - è possibile ispezionarne il contenuto usando il disassemblatore ildasm.exe

>> ildasm.exe mscorlib.dll

Compilazione e Collegamento

- Collegamento ed esecuzione
  - eseguendo il comando Circonferenze.exe, Windows riconosce l'assembly .NET
  - che non contiene codice nativo, ma codice intermedio
  - affida al Framework l'esecuzione (con la compilazione JIT)
  - ed il collegamento dinamico dei namespace esterni utilizzati nel codice

Compilazione e Collegamento



Modificatori di Visibilità

- Riassumendo
  - ⇨in .NET non esiste il concetto di package
- In Java
  - ⇨package = spazio di nomi + cartella di file del disco (spesso contenuto in un .jar)
- In .NET
  - ⇨namespace = spazio di nomi
  - ⇨assembly = gruppo di classi

Modificatori di Visibilità

- E' possibile dire che
  - ⇨i package di Java sono in parte analoghi ai namespace di .NET >> sistema di nomi
  - ⇨i file .jar di java sono in parte analoghi agli assembly di .NET >> archivio di classi
- Livelli di visibilità di C#
  - ⇨public, private e "internal"
  - ⇨non esiste il livello "friendly"

Modificatori di Visibilità

- Il livello internal
  - ⇨visibile in tutte le classi che appartengono allo stesso assembly
  - ⇨non negli altri assembly (anche se collegati)
- Attenzione
  - ⇨internal deve essere specificato esplicitamente (a differenza di "friendly")
  - ⇨es: internal class Prova { ... }

Modificatori di Visibilità

- E se il modificatore manca ?
  - ⇨per le classi il livello standard è public
  - ⇨per i membri delle classi (campi, proprietà, metodi) il livello standard è private
- Di conseguenza
  - ⇨per evitare confusione ed errori, in C# è opportuno sempre specificare esplicitamente il modificatore

Commenti di Documentazione

- Come per Java
  - ⇨anche .NET fornisce una sintassi per immergere commenti di documentazione nel codice
  - ⇨ed uno strumento per generare automaticamente i file di documentazioni
- L'approccio di .NET
  - ⇨basato su XML (>>)

Commenti di Documentazione

>> utilita\Console.cs

- XML
  - ⇨sintassi standard utilizzata in molti contesti
  - ⇨i dati sono codificati con “tag di marcatura”
  - ⇨es: <summary>Descrizione della classe</summary>
- In C#
  - ⇨i commenti di documentazione sono annunciati da ///
  - ⇨e devono contenere dati in formato XML

Commenti di Documentazione

- I principali tag di documentazione
  - ⇨<summary>: descrizione testuale di una classe, un metodo, un campo, una proprietà
  - ⇨<remarks>: annotazioni aggiuntive
  - ⇨<returns>: valore di ritorno di un metodo
  - ⇨<param>: parametro di un metodo

Commenti di Documentazione

>> utilita\Console.cs

- Estrarre i commenti
  - ⇨i commenti vengono estratti in un file con estensione .xml
  - ⇨compito del compilatore, csc.exe
- Opzione /doc:<nomeFile>.xml
  - ⇨es: csc /target:library /doc:Console.xml Console.cs
  - ⇨viene prodotto il file Console.xml
  - ⇨è necessario trasformarlo per poterlo consultare

NDoc

- Il vero analogo di JavaDoc
  - ⇨NDoc – http://ndoc.sourceforge.net
  - ⇨progetto open souce per .NET
  - ⇨uno strumento per generare documentazione leggibile a partire dal file XML generato dal compilatore
- Esempio
  - ⇨consente di generare documentazione nel formato MSDN (formato standard delle API)

NDoc

- L'approccio di NDoc
  - ⇨prende in input un assembly
  - ⇨ed il relativo file di documentazione .xml generato dal compilatore
  - ⇨e produce documentazione in vari formati
- Per ciascun formato
  - ⇨si appoggia ad uno strumento esterno, detto "documenter", incaricato di produrre concretamente la documentazione

NDoc

- Il documenter standard
  - ⇨documenter per il formato MSDN
  - ⇨Microsoft HTML Workshop
  - ⇨scaricabile da http://msdn.microsoft.com
- Il formato MSDN
  - ⇨formato .chm, ovvero HTML compilato ("compiled html")
  - ⇨formato proprietario della Microsoft

NDoc

- Procedura per utilizzare NDoc
  - ⇨scaricare ed installare Microsoft HTML Workshop
  - ⇨scaricare ed installare NDoc
  - ⇨compilare l'assembly e generare la documentazione in formato XML con csc
  - ⇨eseguire NDocGui.exe
  - ⇨specificare l'assembly, il file .xml, il tipo di formato e la cartella destinazione

Convenzioni di Stile di C#

- Le convenzioni suggerite dalla Microsoft
  - ⇨sono leggermente diverse da quelle standard di Java che abbiamo adottato nel corso
- Identificatori
  - ⇨in C# vengono utilizzate due convenzioni diverse
  - ⇨una per gli identificatori pubblici
  - ⇨una per gli identificatori privati

Convenzioni di Stile di C#

- Identificatori privati
  - ⇨campi delle classi, parametri, variabili locali
  - ⇨convenzione cammello (es: raggioCerchio)
- Identificatori pubblici
  - ⇨namespace, classi, metodi, proprietà, costanti
  - ⇨convenzione “Pascal” (discende da Delphi)
  - ⇨convenzione cammello con l’iniziale maiuscola (es: RaggioCerchio)

Convenzioni di Stile di C#

- Quindi, varie differenze significative
  - ⇨nome dei namespace (iniziale maiuscola)
  - ⇨nome dei metodi (iniziale maiuscola)
  - ⇨nome delle costanti (convenzione Pascal)
- Namespace
  - ⇨convenzione suggerita dalla Microsoft
  - ⇨<nomeOrganizz>.<nomeApplicazione>...
  - ⇨es: Unibas.MorraCinese.Modello

Convenzioni di Stile di C#

- Una nota sull'organizzazione dei file
  - ⇨in C# non ci sono vincoli particolari sulla distribuzione del codice nei file
- Ma la convenzione prevede che
  - ⇨a ciascuna classe sia attribuito un file .cs
  - ⇨a ciascun namespace sia attribuita una cartella
  - ⇨es: Unibas.Utilita
  - ⇨es: Unibas.MorraCinese -> Modello

Convenzioni di Stile di C#

- Campi e Proprietà
  - ⇨tipicamente il campo è privato e in notazione cammello; es: private double ascissaCentro;
  - ⇨e la proprietà è pubblica e in notazione Pascal; es: public double AscissaCentro {...}
- Quando usare una proprietà ?
  - ⇨quando il campo deve essere almeno letto (altrimenti usare solo setXXX)

Convenzioni di Stile di C#

- Posizione delle parentesi
  - ⇨non esiste una vera regola
- Negli esempi delle API
  - ⇨posizione alla Java
- Visual Studio
  - ⇨viceversa usa la posizione sempre a capo
  - ⇨sia per le classi che per i metodi che per i blocchi di istruzioni

Convenzioni di Stile di C#

- Nota sulle convenzioni di stile
  - ⇨è opportuno conoscere le convenzioni di stile per un certo linguaggio
  - ⇨ma non è indispensabile adeguarsi completamente
  - ⇨questo andrebbe contro il concetto stesso di stile di scrittura del codice

Riassumendo

- Il Compilatore
  - ⇨Il Concetto di Assembly
  - ⇨Compilazione e Collegamento
- Modificatori di Visibilità
- Commenti di Documentazione
  - ⇨NDoc
- Convenzioni di Stile di C#

Termini della Licenza