CLILibrary

martedì 16 novembre 2010
10.59

Funzionamento di massima

Invocazione metodo

1. Cli_call_3(...):
   Cerca il metodo applicando le convenzioni oppure così com'è e poi lo invoca.

Utilizzo proprietà o campi pubblici

1. Cli_call_3(...):
2. Cli_set_3(...):
   Cerca il campo pubblico o la proprietà e setta i valore passato
3. Cli_get_3(...):
   Cerca il campo pubblico o la proprietà e prende il valore

Problemi

Descrizione: non si riesce ad utilizzare property o campi pubblici statici.
Metodo: *private bool cli_get(Term objId, Term memberTerm, Term what)*
Istruzione: *value = Convert.ToInt32(cl.InvokeMember(memberName, BindingFlags.GetProperty | BindingFlags.Static,Type.DefaultBinder, null,new Object[0]));*
Considerazioni: strano modo di utilizzare una property statica, **non funziona sul framework 3.5**, basta fare così: *value = Convert.ToInt32(property.GetValue(null,null));*
Stato: risolto.

Stesso discorso sopra per i campi statici sia in get che in set (*cli_set)* .

Convention CSharp

OK.

Convention VB.NET

OK.

Convention FSharp

Non supporta le proprietà (forse all'epoca non cerano) e sbaglia a creare il nome dei metodi dato che li mette con la minuscola invece che con la maiuscola.
AGGIUNTO SUPPORTO PER LE PROPERTY E SISTEMATA LA CONVERSIONE PER I NOMI DEI METODI.
OK.

Convention Eiffel.NET

Non funziona perché il framework .NET non riesce a caricare l'assembly (dice che ha un formato sbagliato).
Ho compilato la convenzione con l'ultima versione di EiffelStudio e provato diverse impostazioni, ma sempre stesso problema.
NO.

Convention JSharp

Ci sono problemi con la classe di esempio, sembra che non ci sia il codice ma sole le firme.
Dato che ormai questo linguaggio non è più supportato dalle nuove versioni di visual studio (servirebbe la 2005) ho deciso di rimandare i test su questo linguaggio ad un altro momento (magari se mi rimane un po' di tempo).

Convention IronPython

Come è stata realizzata non può funzionare dato che come scritto nella tesi di Albertin ogni metodo accetta come primo parametro un'istanza di *FunctionEnvironment16Dictionary* e nell'implementazione attuale della convenzione viene passato come valore *null*, quindi il codice che utilizza questo parametro non potrà funzionare e lancerà un'eccezione.
Ecco perché risulta impossibile invocare il costruttore con parametri oppure un qualsiasi metodo.
Anche se dall'IDE di tuProlog si vede un errore di metodo non trovato, in realtà il metodo viene trovato ma l'invocazione lancia eccezione a questo punto il sistema pensa che sia il metodo sbagliato e tenta di trovare quello giusto che ovviamente non trova dato che era già giusto quello prima e ritorna un errore.
Ho provato anche a passargli un istanza vuota (creata con il costruttore di default) che ovviamente non istanziando gli attributi opportuni porta allo stesso problema.
NO.

Considerazioni

Non è un grosso problema se non funzionano le convenzioni per Eiffel, J# e IronPython dato che ormai non sono così utilizzati.
La cosa importante è che funzionino le convenzioni per i principali linguaggi .NET, quali, C#, F# e VB.NET. Le convenzioni per questi linguaggi potrebbero anche essere raccolte in una unica dato che tali linguaggi utilizzano le stesse convenzioni (giro di parole).