Programowanie pod Windows
Zestaw 4

.NET ⇔ Win32
Platform Invoke
COM Interoperability

ogłoszenie listy: 15-03-2005
ważność listy: 05-04-2005

Streszczenie

Możliwości platformy .NET byłyby mocno ograniczone, gdyby niemożliwa była współpraca z kodem niezarządzanym. Podobnie jednak jak istnieją dwa różne typy niezarządzanych bibliotek, bibilioteki natywne i biblioteki COM, tak istnieją dwa różne mechanizmy do współpracy z nimi, **Platform Invoke** do konsumpcji bibliotek natywnych oraz **COM Interoperability** do konsumpcji i produkcji usług COM.

Współpraca z już istniejącym kodem niezarządzanym oznacza tak naprawdę możliwość stopniowego wprowadzania platformy .NET do już istniejących projektów, bez konieczności kosztownego jednorazowego przenoszenia ich do .NET w całości. To również szansa na wspołpracę .NET z technologiami, które z jakichś powodów nigdy nie zostaną przeniesione do kodu zarządzanego.

Poniższe zadania umożliwią dokładne poznanie obu mechanizmów współpracy z kodem niezarządzanym.

1. **P/Invoke, Win32 ⇒ .NET**

   Napisać w C# program, w którym zostanie wywołana funkcja Win32 `GetUserName`, a jej wynik zostanie wyprowadzony w oknie informacyjnym, wywołanym przez funkcję Win32 `MessageBox`.

   Wskazówka: użyć atrybutów `DllImport`*, zadeklarować obie funkcje jako* `extern`*.*

   [**1p**]

2. **P/Invoke + DLL**

   Napisać w języku C bibliotekę natywną, która udostępnia funkcję `int IsPrimeC`, sprawdzającą czy podana 32-bitowa liczba jest pierwsza.

   Napisać program w C#, który wywoła tę funkcję z parametrem podanym przez użytkownika z konsoli.

   [**1p**]

3. **P/Invoke + DLL + wskaźniki na funkcje/delegacje**

   Napisać w języku C bibliotekę natywną, która udostępnia funkcję `int ExecuteC` przyjmującą dwa parametry: 32-bitową wartość `n` i wskaźnik na funkcję o sygnaturze `int f(int)`. Funkcja `Execute` jako wynik powinna zwracać wartość `f(n)`.

   Napisać program w C#, który oprócz funkcji `Main` będzie zawierał funkcję `int IsPrimeCs` i który użyje funkcji `ExecuteC` (zastosowanej do funkcji `IsPrimeCs`) do sprawdzenia czy podana przez użytkownika z konsoli liczba jest pierwsza.

   Czy możliwe było przeniesienie kodu funkcji `IsPrimeC` z poprzedniego zadania jako funkcji `IsPrimeCs`?

   [**2p**]

4. **COM Interop, COM ⇒ .NET, early/late binding**

   To zadanie składa się z 3 części:

   (a) Napisać bibilotekę COM, która będzie zawierała klasę `PrimeTester`, a w niej metodę `int IsPrime`. Napisać skrypt powłoki, w którym ta metoda zostanie wywołana, a wynik pokazany w oknie informacyjnym.
   Wskazówka: tworzenie bibliotek COM zostało omówione na wykładzie. Zastosować zaproponowaną tam metodę: projekt C++ typu **ATL Library**, *do niego dodana klasa* **ATL COM+ 1.0 Component**.

   (b) Napisać program w C#, w którym zostanie wywołana funkcja `IsPrime` z poprzedniego zadania. Użyć klasy opakowującej (utworzonej automatycznie lub ręcznie).

   (c) Napisać program w C#, w którym zostanie wywołana funkcja `IsPrime` z poprzedniego zadania. Zamiast klasy opakowującej użyć refleksji.

   Jakie są wady i zalety wczesnego i późnego wiązania (łatwość użycia, bezpieczne typowanie)? Czy użycie wczesnego wiązania jest zawsze możliwe?

   Wskazówka: nauczyć się korzystać z `regsvr32.exe` *do rejestrowania i wyrejestrowywania komponentów COM. Nauczyć się korzystać z* `tlbimp.exe` *do tworzenia klas .NET opakowujących klasy COM.*

   [**2p**]

5. **COM Interop, .NET ⇒ COM**

   Napisać w C# bibliotekę, która będzie zawierała klasę `PrimeTesterCS`, a w niej metodę `int IsPrime`. Zarejestrować tę bibliotekę jak bibliotekę COM. Napisać w C++ niezarządzanego klienta COM, zwykłą aplikację konsoli, która skorzysta z tej biblioteki.

   Jakie warunki muszą być spełnione, aby klasa .NET mogła być zarejestrowana jako biblioteka COM?

   Wskazówki:

   (a) Nauczyć się korzystać z atrybutu `GuidAttribute`. *Dlaczego warto użyć go do oznaczenia klasy* `PrimeTesterCS`*? Co stałoby się, gdyby nie został on użyty?*

   (b) Nauczyć się korzystać z `sn.exe` *do tworzenia plików z sygnaturami cyfrowymi. Silnie cyfrowo osygnować bibliotekę, umieszczając odpowiedni atrybut w* `AssemblyInfo.cs`. *Dlaczego trzeba silnie sygnować biblioteki przeznaczone do COM Interop?*

   (c) Nauczyć się korzystać z `gacutil.exe` *do zarządzania GAC. Dodać bibliotekę do GAC.*

   (d) Nauczyć się korzystać z `regasm.exe` *do rejstrowania bibliotek .NET jako komponentów COM. Przy okazji obejrzeć efekt działania* `regasm.exe` *z parametrem* `/regfile`. *Zarejestrować bibliotekę dla COM Interop.*

   (e) Nauczyć się korzystać z `tlbexp.exe` *do eksportowania informacji z bibliotek .NET do współpracy z COM. Dlaczego trzeba eksportować informacje o typach do pliku* `*.tlb` *(TypeLiB)?*

   (f) Nauczyć się korzystać z dyrektywy `#import` *do tworzenia klientów COM w niezarządzanym C++. Dlaczego dyrektywy tej należy użyć wskazując jako parametr ścieżkę do pliku* `*.tlb`, *a nie do biblioteki* `*.dll`*?*

   Uwaga! Ze względu na pewną trudność zadania, za częściowe rozwiązania będą wyjątkowo przyznawane punkty pośrednie (między 1 a 4).

   [**4p**]