

在 Visual c + + 中從 System :: 字串轉換成 Char

2020/10/10 •  

本文內容

[摘要](#)

[方法 1](#)

[方法 2](#)

[方法 3](#)

[方法 4](#)

[Visual c + + 2002 或 Visual c + + 2003 \(的 c + + Managed Extension 範例程式碼\)](#)

本文說明 `System::String* char*` 使用 Visual c + + 中的受管理擴充轉換的幾種方式。

原始產品版本： Visual c + +

原始 KB 編號： 311259

摘要

本文是指下列 Microsoft .NET Framework 類別庫命名空間：

- `System::Runtime::InteropServices`
- `Mscrl::interop`

本文將討論下列幾種使用下列方式轉換成的方式 `System::String* char*`：

- Visual c + + .net 2002 和 Visual c + + .NET 2003 中 c + + 的 Managed extensions
- Visual c + + 2005 和 Visual c + + 2008 中的 c + + /CLI

方法 1

`PtrToStringChars` 會提供實際物件的內部指標 `String`。如果您將此指標傳遞給非受管理的函數呼叫，您必須先鎖定指標，以確保物件在非同步垃圾回收過程中不會移動：

C++


 複製

```
//#include <vcclr.h>
System::String * str = S"Hello world\n";
const __wchar_t __pin * str1 = PtrToStringChars(str);
wprintf(str1);
```

方法 2

`StringToHGlobalAnsi` 將受管理物件的內容複製到本機堆，然後將其轉換為美國國家標準協會 (ANSI) 格式。此方法會將所需的原生堆記憶體分配給您：

C++

 複製

```
//using namespace System::Runtime::InteropServices;  
System::String * str = S"Hello world\n";  
char* str2 = (char*)(void*)Marshal::StringToHGlobalAnsi(str);  
printf(str2);  
Marshal::FreeHGlobal(str2);
```

ⓘ 注意

在 Visual c + + 2005 和 Visual c + + 2008 中，您必須將公共語言執行時間支援編譯器選項 (/clr : oldSyntax) 中，以順利編譯先前的程式碼範例。若要新增公共語言執行時間支援編譯器選項，請遵循下列步驟：

1. 按一下 [專案]，然後按一下 [ProjectName 屬性]。

ⓘ 注意

ProjectName 是專案名稱的預留位置。

2. 展開 [設定 屬性]，然後按一下 [一般]。
3. 在右窗格中，按一下以選取 [**公用語言執行時間支援]，舊的語法 (/clr : oldSyntax) ** 在 通用語言執行時間支援 專案設定。
4. 在 [撥號對應表 (電話內容)] 方塊中，按一下 [瀏覽] 以尋找使用者的撥號對應表。

如需有關常見語言執行時間支援編譯器選項的詳細資訊，請造訪下列 Microsoft Developer Network (MSDN) 網站：


[/clr \(常見語言執行時間編譯\)](#)

這些步驟適用於整篇文章。

方法 3

VC7 `CString` 類別具有一個採用 `Managed` 字串指標的建構函式，並 `CString` 以其內容載入：

C++


 複製

```
//#include <atlstr.h>
System::String * str = S"Hello world\n";
CString str3(str);
printf(str3);
```

方法 4

Visual c++ 2008 引進 `marshal_as<T>` 封送處理說明類別和 `marshal_context()` 封送協助程式類別。

C++

 複製


```
//#include <msclr/marshal.h>
//using namespace msclr::interop;
marshal_context ^ context = gcnew marshal_context();
const char* str4 = context->marshal_as<const char*>(str);
puts(str4);
delete context;
```

ⓘ 注意

在 Visual c++ .net 2002 或 Visual c++ .NET 2003 中使用 c++ 的 managed extensions 時，不會編譯此程式碼。它使用 Visual c++ 2005 中引進的新 c++/CLI 語法，以及 Visual c++ 2008 中引進的新 msclr 命名空間碼。若要順利編譯此程式碼，您必須使用 Visual c++ 2008 中的 `/clr c++` 編譯器參數。

Visual c++ 2002 或 Visual c++ 2003 (的 c++ Managed Extension 範例程式碼)

C++

 複製

```
//compiler option: cl /clr
#include <vcclr.h>
#include <atlstr.h>
#include <stdio.h>
using <mscorlib.dll>
using namespace System;
using namespace System::Runtime::InteropServices;

int _tmain(void)
{
    System::String * str = S"Hello world\n";
```

```
//method 1
const __wchar_t __pin * str1 = PtrToStringChars(str);
wprintf(str1);


//method 2
char* str2 = (char*)(void*)Marshal::StringToHGlobalAnsi(str);
printf(str2);
Marshal::FreeHGlobal(str2);

//method 3
CString str3(str);
wprintf(str3);

return 0;
}
```

Visual c + + 2005 和 Visual c + + 2008 (的 c + +/CLI 範例程式碼)

C++

 複製

```
//compiler option: cl /clr

#include <atlstr.h>
#include <stdio.h>
using <mscorlib.dll>

using namespace System;
using namespace System::Runtime::InteropServices;

#if _MSC_VER > 1499 // Visual C++ 2008 only
#include <msclr/marshal.h>
using namespace msclr::interop;
#endif

int _tmain(void)
{
    System::String ^ str = "Hello world\n";

    //method 1
    pin_ptr<const wchar_t> str1 = PtrToStringChars(str);
    wprintf(str1);

    //method 2
    char* str2 = (char*)Marshal::StringToHGlobalAnsi(str).ToPointer();
    printf(str2);
    Marshal::FreeHGlobal((IntPtr)str2);

    //method 3
    CString str3(str);
}
```

```
wprintf(str3);

//method 4
#if _MSC_VER > 1499 // Visual C++ 2008 only
marshal_context ^ context = gcnew marshal_context();
const char* str4 = context->marshal_as<const char*>(str);
puts(str4);
delete context;
#endif

return 0;
}
```

此頁面有所助益嗎？

 Yes  No
