

# View the type of a C# generic in the debugger



14



1

When I hover over a generic type in Visual Studio using the debugger, I don't get the current type, is there a way to display it without going to the immediate window and typing `?typeof(T).Name` ?

c#

visual-studio

edited Sep 22 '17 at 12:22



French Boiethios

12.4k 4 44 87

asked Jan 6 '16 at 8:29



user4388177

918 1 9 18

What programming language? – [Ivan Aksamentov - Drop](#) Jan 6 '16 at 8:46

C# xxxxxxxxxxxx – [user4388177](#) Jan 6 '16 at 8:54

Then I guessed wrong ;) – [Ivan Aksamentov - Drop](#) Jan 6 '16 at 8:54

Yeah, I thought typeof was self-explanatory, but I completely forgot CLR C++ :) – [user4388177](#) Jan 6 '16 at 8:56

There are `typeid` and `typeof` even in native C++ – [Ivan Aksamentov - Drop](#) Jan 6 '16 at 8:57

## 3 Answers



15



You can see the types in the callstack window by looking at the top line which will show the runtime evaluated type.

Also want to emphasize for others your suggestion:

going to the immediate window and typing `?typeof(T).Name`

Join Stack Overflow to learn, share knowledge, and build your career.

[Sign up with email](#)[Sign up with Google](#)[Sign up with Facebook](#)

"You can see the types in the callstack window by looking at the top line which will show the runtime evaluated type" -> IF you enable "show parameter value" – [kofifus](#) Aug 2 '18 at 1:17

You can see full types of variables in watch windows such as "watch", "autos", "locals". Also, you can enable types in call stack window (in right-click context menu).

4

Here is an example for C++ (works for C# the same way):

The screenshot shows the Visual Studio IDE with a C++ file named `main.cpp` open. The code defines a template `MyType` and a function `foo`. The `foo` function takes two iterators, `first` and `last`, and returns a `MyType` object. The `main` function calls `foo` with iterators from a `std::vector`.

The **Locals** window shows the following variables:

Name	Value	Type
<code>first</code>	<code>{1}</code>	<code>std::_Vector_iterator&lt;std::_Vector_val&lt;std::_Simple_types&lt;int&gt; &gt; &gt;</code>
<code>[ptr]</code>	<code>0x000001cf9cc55540 {1}</code>	<code>int *</code>
<code>[Raw View]</code>	<code>{...}</code>	<code>std::_Vector_iterator&lt;std::_Vector_val&lt;std::_Simple_types&lt;int&gt; &gt; &gt;</code>
<code>last</code>	<code>{-33686019}</code>	<code>std::_Vector_iterator&lt;std::_Vector_val&lt;std::_Simple_types&lt;int&gt; &gt; &gt;</code>
<code>v2</code>	<code>{ size=3 }</code>	<code>std::vector&lt;int, std::allocator&lt;int&gt; &gt;</code>
<code>[capacity]</code>	<code>3</code>	<code>_int64</code>
<code>[allocator]</code>	<code>allocator</code>	<code>std::_Compressed_pair&lt;std::_Wrap_alloc&lt;std::allocator&lt;int&gt; &gt;, std::_Vector</code>
<code>[0]</code>	<code>2</code>	<code>int</code>
<code>[1]</code>	<code>3</code>	<code>int</code>
<code>[2]</code>	<code>4</code>	<code>int</code>
<code>[Raw View]</code>	<code>{...}</code>	<code>std::vector&lt;int, std::allocator&lt;int&gt; &gt;</code>
<code>x</code>	<code>{val=3.14000010 }</code>	<code>MyType&lt;float&gt;</code>
<code>val</code>	<code>3.14000010</code>	<code>float</code>

The **Call Stack** window shows the following frames:

Name	Lang
<code>cppapp164d.exe!foo&lt;std::_Vector_iterator&lt;std::_Vector_val&lt;std::_Simple_types&lt;int&gt; &gt; &gt; &gt;(std::_Vector_iterator&lt;std::_Vector</code>	C++
<code>cppapp164d.exe!main() Line 17</code>	C++
<code>cppapp164d.exe!invoke_main() Line 75</code>	C++
<code>cppapp164d.exe!scr common main seh() Line 264</code>	C++

answered Jan 6 '16 at 8:52



[Ivan Aksamentov - Drop](#)

10.9k 2 26 58

This just works if I have a variable of that type. – [user4388177](#) Jan 6 '16 at 8:54

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email

Sign up with Google

Sign up with Facebook





You can add a watcher for `typeof(T)` .

1



answered Oct 4 '17 at 7:48



[Lars Gyrup Brink Nielsen](#)

2,387 1 18 26

**Join Stack Overflow** to learn, share knowledge, and build your career.

[Sign up with email](#)

 [Sign up with Google](#)

[Sign up with Facebook](#)

