Structured binding in C++/WinRT: The key-value pair

devblogs.microsoft.com/oldnewthing/20201016-00

October 16, 2020

Raymond Chen

Last time, we learned how to add structured binding support to your own types, and noted that the `get` functions do not have to return references.

C++/WinRT has something similar to a `std::pair`: The `IKeyValuePair<K, V>`, which is used to represent a single entry in a map. (C++ uses a `std::pair` for this.)

Since the `kvp.Key()` and `kvp.Value()` methods always return by value, it means that when you use structured binding on an `IKeyValuePair`, the variables are always copies. The qualifiers on the `auto` merely describes how the `kvp` itself is captured.

```
IKeyValuePair<hstring, int32_t> kvp;
auto& [key, value] = kvp;
```

This looks like it's binding `key` and `value` as lvalue references, but it's not. They are non-reference variables. That's because the code expands to

```
auto& hidden = kvp;
decltype(auto) key = kvp.Key();
decltype(auto) value = kvp.Value();
```

Since the results stored into `key` and `value` don't depend on how you bound the source, you may as well bind the hidden variable by reference to avoid an unnecessary copy.

```
// wasteful copy from kvp to hidden
auto [key, value] = kvp;

// non-copying binding
auto&& [key, value] = kvp;
```

Bonus chatter: The structured binding of `IKeyValuePair` comes in particularly handy when you are iterating over something like an `IMap`:

```
for (auto&& [key, value] : map)
{
  // use key and value
}
```

Raymond Chen

Follow