How do I set multiple items to a Windows Runtime vector in a single call?

devblogs.microsoft.com/oldnewthing/20200727-00

July 27, 2020

Raymond Chen

Suppose you want to set multiple items to a Windows Runtime `IVector<T>`. This is common in the cases where the system provides a vector that you are expected to fill with stuff. For example:

```
// C#
var picker = new FileOpenPicker();
picker.FileTypeFilter.Add(".bmp");
picker.FileTypeFilter.Add(".gif");
picker.FileTypeFilter.Add(".jpg");
picker.FileTypeFilter.Add(".png");
```

Surely there is an easier way to do this than calling `Add` multiple times, right?

Yes, there is an easier way, but the easier way depends on what language you are using. Each language expresses the Windows Runtime `IVector<T>` in its own language-specific way.

C# projects the `IVector<T>` as an `System.Collections.Generic.IList<T>`. You can use object and collection initializer syntax to fill the collection as part of the object initialization.

```
// C#
var picker = new FileOpenPicker()
    {
        FileTypeFilter = { ".bmp", ".gif", ".jpg", ".png" }
    };
```

Note, however, that this syntax works only in an object initializer.

```
// doesn't work
picker.FileTypeFilter = { ".bmp", ".gif", ".jpg", ".png" };
```

You might be tempted to use `List<T>.AddRange()`, but that doesn't work either because what you have is an `IList<T>`, not a `List<T>`. Many people have solved this problem by using an extension method.

C++/WinRT exposes the `IVector<T>` very close to how it is defined in the ABI. In particular, there is a `ReplaceAll` method.

```
// C++/WinRT
auto picker = FileOpenPicker();
picker.FileTypeFilter().
    ReplaceAll({ L".bmp", L".gif", L".jpg", L".png" });
```

C++/CX is a bit more annoying because you have to pass a `Platform:: Array^` to the `ReplaceAll` method, and those `Array^` types are frustrating to manufacture.

```
// C++/CX
auto picker = ref new FileOpenPicker();
String^ extensions[]{ L".bmp", L".gif", L".jpg", L".png" };
picker->FileTypeFilter->ReplaceAll(
    ArrayReference<String^>(extensions, _ARRAYSIZE(extensions)));
```

Sadly, there are no deduction guides for `ArrayReference` so you end up having to repeat `String^`.

JavaScript projects `IVector<T>` as a native JavaScript `Array`, and those objects have quite a rich panoply of available methods. One that is useful for us today is `splice`.

```
// JavaScript
var picker = new Windows.Storage.Pickers.FileOpenPicker();
picker.fileTypeFilter.splice(0, 0, ".bmp", ".gif", ".jpg", ".png");
```

The JavaScript projection is kind enough to project the original `replaceAll` method as well, which leads us to this somewhat simpler version:

```
// JavaScript
var picker = new Windows.Storage.Pickers.FileOpenPicker();
picker.fileTypeFilter.replaceAll([ ".bmp", ".gif", ".jpg", ".png" ]);
```

Raymond Chen

Follow