What's the difference between DataPackageView.GetUri-Async and DataPackageView.GetWebLinkAsync?

devblogs.microsoft.com/oldnewthing/20240801-00

August 1, 2024

Raymond Chen

The Windows Runtime `DataPackage` object has methods for manipulating three types of URIs:

`StandardDataFormats` value	`DataPackage` method	`DataPackageView` method
`Uri`	`SetUri`	`GetUriAsync`
`WebLink`	`SetWebLink`	`GetWebLinkAsync`
`ApplicationLink`	`SetApplicationLink`	`GetApplicationLinkAsync`

What do the three different URIs mean, and how do they differ?

Once upon a time, there was only one URI data format. And it was called `Uri`.

`StandardDataFormats` value	`DataPackage` method	`DataPackageView` method
`Uri`	`SetUri`	`GetUriAsync`

Windows 8.1 added a second URI data format called `ApplicationLink`, so that apps could add a link that relaunched the app to return to the item that was copied. For example, if the Contoso app copies a customer service record to the clipboard, it can add an `Application-Link` that is a link into the Contoso app that navigates to that customer service record.

Since we now had two URI data formats, it was confusing to have a data format named simply `Uri`, so the old `Uri` format was renamed to `WebLink`.

Data format	Meaning
`WebLink`	Link to Web resource.
`ApplicationLink`	Link to app to view the item.

For backward compatibility, we still have to support the old unfashionable API, but `Uri` is just an alternate name for `WebLink`. The `Uri` data format is identical to the the `WebLink` data format. The `SetUri` method does exactly the same thing as the `SetWebLink` method. The `GetUriAsync` method does exactly the same thing as the `GetWebLinkAsync` method.

For example, if an app uses `SetUri` to set a URI, and you then call `GetUriAsync`, it will produce that same URI. The `Uri` and `WebLink` are literally the same thing.

Our final table therefore is

`StandardDataFormats` value	`DataPackage` method	`DataPackageView` method
`Uri` `WebLink`	`SetUri` `SetWebLink`	`GetUriAsync` `GetWebLinkAsync`
`ApplicationLink`	`SetApplicationLink`	`GetApplicationLinkAsync`

The fact that `Uri` and `WebLink` are identical means that your program doesn't have to try to handle both. Just decide which name you want to use for the format (either `Uri`, the OG name; or `WebLink`, the hip new name), and use it.

```
namespace winrt
{
    using namespace winrt::Windows::Foundation::Uri;
    using namespace winrt::Windows::ApplicationModel::DataTransfer;
}

winrt::Uri TryGetUri(winrt::DataPackageView const& view)
{
    if (view.Contains(StandardDataFormats::ApplicationLink())) {
        return co_await dataPackageView.GetApplicationLinkAsync();
    } else if (view.Contains(StandardDataFormats::WebLink())) {
        return co_await dataPackageView.GetWebLinkAsync();
    } else if (view.Contains(StandardDataFormats::Uri())) {
        return co_await dataPackageView.GetUriAsync();
    }
    return nullptr;
}
```

The above example decides that it wants to prefer the application link (which takes the user back to the app that provided the data package), and if that's not available, then it sees if the data package contains a Web link (to view the content in a Web browser), and if even that's not available, then it looks for a Uri (also to view the content in a Web browser).

But the last test is redundant because `WebLink` and `Uri` are the same thing. If a `Uri` is present, then `Contains(WebLink)` will find it. The test for `Uri` is dead code.

It's like taking attendance in a class, and there's a student whose name is Joseph, but he also uses the nickname Joe. If you ask, "Is Joseph here?", and there is no answer, then there's no point asking, "Is Joe here?" because Joe and Joseph are the same person. There will never be a response to "Is Joe here?"

So once we know that Joseph isn't in the data package, there's no point asking if Joe is in it.

```
winrt::Uri TryGetUri(winrt::DataPackageView const& view)
{
    if (view.Contains(StandardDataFormats::ApplicationLink())) {
        return co_await dataPackageView.GetApplicationLinkAsync();
    } else if (view.Contains(StandardDataFormats::WebLink())) {
        return co_await dataPackageView.GetWebLinkAsync();
    // } else if (view.Contains(StandardDataFormats::Uri())) {
    //     return co_await dataPackageView.GetUriAsync();
    }
    return nullptr;
}
```