From a Windows app, how can I check whether there is an app installed that implements a particular URI scheme?, part 2

devblogs.microsoft.com/oldnewthing/20230309-00

March 9, 2023

Raymond Chen

Last time, we looked at detecting packaged apps which support a particular URI scheme. Unpackaged apps do not have AppIds (in the Windows Store sense), so some of the operations don't work.

Function	Packaged apps	Unpackaged apps
`Launcher.FindUriSchemeHandlersAsync`	Yes	No
`Launcher.QueryUriSupportAsync(uri)`	Yes	Yes
`Launcher.QueryUriSupportAsync(uri, pfn)`	Yes	N/A
`Launcher.LaunchUriAsync`	Yes	Yes

If you think about it, the reasons for the above entries are obvious:

- `Launcher.FindUriSchemeHandlersAsync` returns a collection of `AppInfo` objects, and `AppInfo` objects describe packaged apps. So it has no way to report an unpackaged app.
- `Launcher.QueryUriSupportAsync(uri)` just tells you whether the URI can be launched or not. If it can be launched by an unpackaged app, then it will report `Available`, just like the case where it can be launched by a packaged app. It doesn't tell you *which* app will launch it, so it doesn't run into the problem of trying to describe something that it has no way to describe.
- `Launcher.QueryUriSupportAsync(uri, pfn)` takes a package family name, and unpackaged apps don't have a package family name, so it's not even possible to specify the unpackaged app you are querying for.

- `Launcher.LaunchUriAsync` tries to launch the URI and tells you whether it succeeded. It doesn't tell you anything about the app that ultimately handled the URI, so unpackaged apps don't cause any problems.

But what if you want to ask about unpackaged apps, too?

The `SHAssocEnumHandlersForProtocolByApplication` function gives you the apps (both packaged and unpackaged) which can launch a particular URI scheme.

Today's smart pointer library will be (rolls dice)[1] WRL.

```
Microsoft::WRL::ComPtr<IEnumAssocHandlers> e;
auto hr = SHAssocEnumHandlersForProtocolByApplication(L"http", IID_PPV_ARGS(&e));
if (SUCCEEDED(hr)) {
    Microsoft::WRL::ComPtr<IAssocHandler> handler;
    while (e->Next(1, &handler, nullptr) == S_OK) {
        PWSTR name;
        if (SUCCEEDED(handler->GetUIName(&name))) {
            printf("UI Name: %ls\n", name);
            CoTaskMemFree(name);
        }
    }
}
```

You can even ask pass the URI to a specific handler by calling the `Invoke` method:

```
HRESULT InvokeHandlerOnURI(IAssocHandler* handler, PCWSTR uri)
{
    Microsoft::WRL::ComPtr<IShellItem> item;
    RETURN_IF_FAILED(SHCreateItemFromParsingName(
        L"http://msn.com/", nullptr, IID_PPV_ARGS(&item)));
    Microsoft::WRL::ComPtr<IDataObject> dto;
    RETURN_IF_FAILED(item->BindToHandler(nullptr,
        BHID_DataObject, IID_PPV_ARGS(&dto)));
    RETURN_IF_FAILED(handler->Invoke(dto.Get()));
    return S_OK;
}
```

[1] Dirty secret: The dice are loaded.