Project Structure

Overview

`PowerToys Run` is divided across several projects to keep a logical separation between plugins and core functionality. The following sections provide a brief overview of each project.

Image of project dependency Fig 1. Project along with their dependencies in `PowerToys Run` ecosystem.

Project Description

`PowerLauncher`

This is the startup project for the `PowerToys Run.` It is a WPF desktop application and follows the `Model-View-ViewModel (MVVM)` design pattern. Plugins play the role of `Model` and provide data to `ViewModel.`

`PowerLauncher.Telemetry`

`PowerLauncher.Telemetry` is a .net core project that contains telemetry events generated by `PowerLauncher.` These events have been discussed in detail here.

`Wox.Core`

`Wox.Core` is a .net core project that contains helper classes required by the `PowerLauncher` project. Two major functionalities encapsulated in this project are `PluginManager` and `Query Builder.` `PluginManager` provides an interface for managing C# plugins. `Query Builder.` decimate user-typed query string and creates a `Query` object. `Query` object contains the action keyword and cleaned query, which is then sent to all plugins.

`Wox.Infrastructure`

`Wox.Infrastructure` is a .net core project that contains helper classes required for image manipulation and storage by the `PowerLauncher` project and the plugins. `ImageLoader.cs` class is used to load icons for `Win32` program. It also provides caching functionality to speed up image loading for frequently queried programs.

`Wox.Plugin`

`Wox.Plugin` contains interfaces that facilitate communication between `PowerLauncher` and plugins. These interfaces have been discussed in detail here. It also contains a helper class for logging. `Log.cs` provides an abstraction for logging error, information, and output to text files. These files are stored at `%userprofile%/appdata/local/microsoft/powertoys/powertoys run/Logs.`