Easy Window Service Documentation

1. BaseOptionWindow.cs

Description: This is a base class that provides functionality for windows that require additional options. It includes methods to show and hide these options, allowing developers to easily extend functionality for different types of windows with custom settings.

Method/Property	Description
ShowOptions()	Displays additional settings or options that are part of the window. This can be used when users need more advanced configurations for the window.
HideOptions()	Hides the additional options, restoring the window to its default, simplified state. Useful when users finish interacting with advanced settings.

2. BaseWindow.cs

Description: This is the foundational class for all window types in the application. It provides the basic structure for opening, closing, and tracking the state of the window (whether it is open or closed). This class can be extended to create different types of windows with custom behavior.

Method/Property	Description
OpenWindow()	Opens the window and makes it visible to the user. Can include animations or transitions, depending on the specific implementation.
CloseWindow()	Closes the window and removes it from view. Typically, any unsaved data should be handled before this method is called.
IsOpen	Returns the current state of the window. True if the window is open, and False if it is closed. This is useful for checking window states in logic or UI updates.

3. IWindow.cs

Description: This interface defines the essential methods that all window objects must implement. It ensures that each window has a consistent set of functions for opening, closing, and visibility tracking, making it easier to manage different window types.

Method/Property	Description
Open()	Defines the logic to open the window. Every window must implement this method, which will be responsible for displaying the window and any associated UI elements.
Close()	Defines the logic to close the window. This method ensures that all windows can be dismissed properly, with any cleanup or save operations occurring here.
IsVisible	A property indicating whether the window is currently visible to the user. Useful for managing UI interactions and performance optimizations when windows are not in use.

4. IWindowsService.cs

Description: This interface defines how the application manages multiple windows. By implementing this service, developers can control the lifecycle of windows, ensuring that they are properly opened, closed, and transitioned between, depending on the application's needs.

Method/Property	Description
ShowWindow(IWindow)	Takes an instance of a window and makes it visible to the user. This method is used to manage the visibility of windows across different parts of the application.
HideWindow(IWindow)	Takes an instance of a window and hides it from view. This method is typically used when switching between windows or when a window is no longer needed.

5. IWindowsServiceCanvas.cs

Description: This interface extends the window management system to work specifically with Unity's Canvas system. It includes methods for displaying and hiding windows on the Canvas, allowing for seamless integration with UI components in Unity-based projects.

Method/Property	Description
ShowOnCanvas(IWindow)	Displays a window on the Unity Canvas. This method ensures that the window is properly rendered within the game's UI system.
HideFromCanvas(IWindow)	Hides a window from the Unity Canvas. Useful for temporarily removing windows from view without destroying their instance.

Sample

Description: The Sample folder contains example implementations demonstrating how to use the EasyWindowsService package in a real-world application. It includes sample windows and configurations that showcase the integration of the IWindow interface and the service for window management.

- **ExampleWindows.cs:** A sample window implementation that shows how to utilize the base window classes and interfaces. This example provides a template for creating new windows with minimal effort.
- **SampleConfigurations:** Provides example configurations for setting up and managing windows within a Canvas or UI framework. This helps developers quickly understand how to apply the package in their own projects.