



























































































































































































































































































































































































































































































































































































Import Dependencies Continued



- Legacy codebases often have their required dependencies hidden in their source code instead of shared header files.
 - Eliminate as many "dead" parameters and imports as possible.
 - Use packages and dynamic libraries (.so/.dlls) to keep code isolated and modular.
 - Consider making a "Testing" specific header/source file that overrides specific tricky-to-mock methods and only implements necessary functions.
 - In some languages, you can also declare classes within functions and other classes for temporary use.

Onion Parameters

Theme

// Hidden for simplicity...

// Hidden for simplicity...

ApplicationWindow

- width: int
 height: int
 Hidden for simplicity...
- +ApplicationWindow(int, int ...): void +handleMouseClick(Event): void +handleKeyboardInput(Event): void // Hidden for simplicity...

BrowserWindow

- mc : MouseCursor - theme : Theme // Hidden for simplicity...
- +BrowserWindow(MouseCursor,int , int, Theme) : void +handleMouseClick(Event) : void +handleKeyboardInput(Event) : void // Hidden for simplicity...

WebBrowser

- engine : BrowserEnginewindow : BrowserWindow// Hidden for simplicity.....
- + WebBrowser
 (BrowserEngine,BrowserWindow): void
 + startup(): void
 + shutdown(): void
 + processRequest(Request): void
 // Hidden for simplicity......

BrowserEngine

- parser : Parser // Hidden for simplicity...
- +BrowserEngine(Parser, Runtime ...) : void // Hidden for simplicity...