Design: Control

Contents:

Introduction

This document provides an overview of the design and requirements of the Paragon Web Browser control, a.k.a. the Control. The control forms the foundation for the Paragon framework.

Requirements

- The control should be embeddable in both windows forms and WPF screens.
- The control should be self-contained and should be embeddable using the normal mechanisms supported by the IDE.
- The control should provide hooks to the container in the form of events, to control most of (if not all) the behavior of the browser.
- The control should support bootstrapping with a standard URLs or a application/workspace package Urls.
- The control should not allow navigation to external resources and should setup default CORS restrictions.
- The control should provide reasonable default implementations for common windows like popup browsers, alerts, prompts etc.
- The control should facilitate the secure loading of the paragon kernel javascript plugins once for each application.
- The control should also facilitate the secure loading of application specific javascript plugins once for each application.
- The control should inject the javascript plugins only into contexts created for application windows.
- Embedded Control
    - HTML5 App Content provided:
        - Stringafied doc
        - url to webserver content
        - url to local embedded package
        - url to app store
    - Native App container interaction only available through a single messaging api to post and consume messages inside the paragon app and outside in the Native app Container.
    - Possible shim library for Ocean as opposed to direct control.
    - Preserve as much of the kernel that makes sense outside the container.
    - Key assumption: Proper paragon is installed on everyone's machine which includes the embedded control.

Design

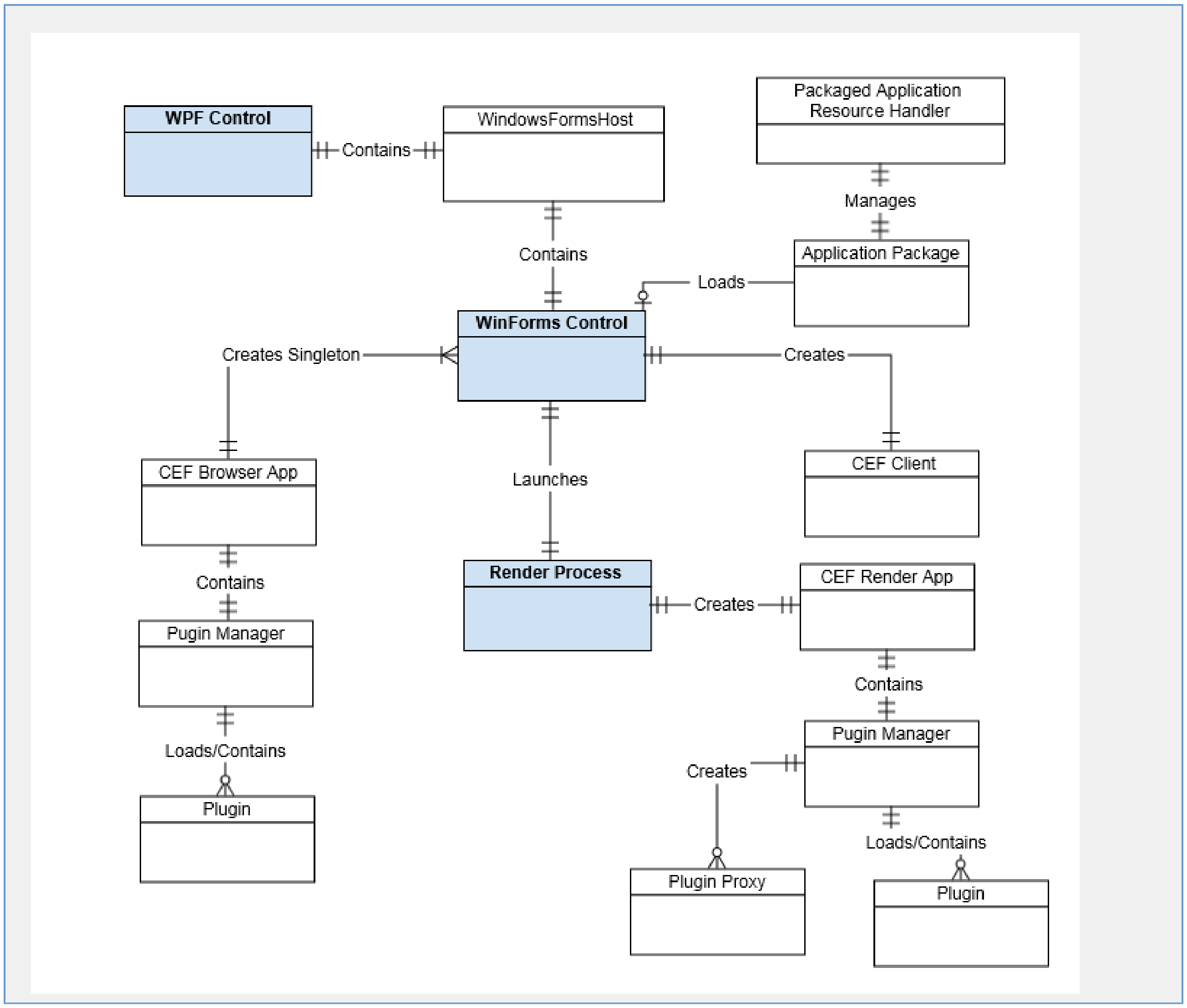
Significant design decisions

- The control will based on CEF3. The new implementation shall be entirely in C# as it would facilitate contribution from a wider range of developers. To facilitate this, the new implementation shall be based on a popular .NET bindings for CEF3 called CefGlue. CefGlue is opensource and can be easily kept up-to-date with CEF3 releases.
- As dictated by the requirements, there shall be two flavors of this control; one for winforms and one for WPF.
- The core of the implementation shall be in the form of a full fledged winforms control.
- A native implementation of the control for WPF (windowless), though feasible leveraging the offscreen rendering capabilities of chromium, is not be desirable because of the performance limitations of that technique. Offscreen rendering as of now, can't leverage the

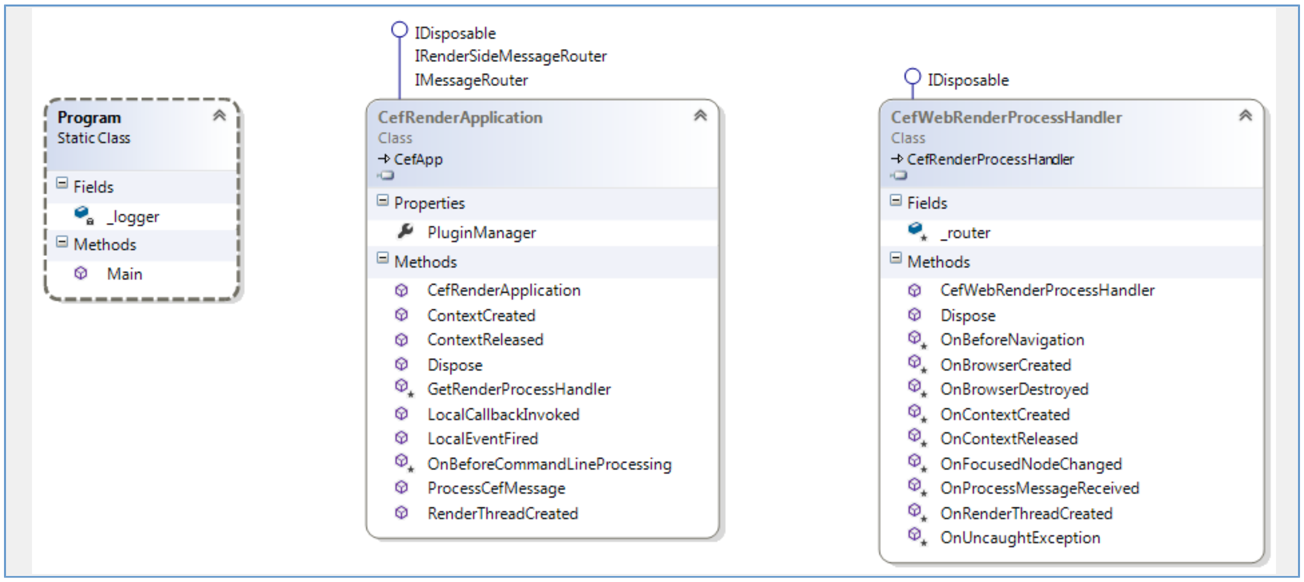
hardware acceleration. Consequently, the new WPF control, will simply wrap the winforms control with a WindowsFormsHost control.

- A decision regarding the choice of chromium process model needs to be further discussed and agreed upon. The process will have a significant impact on the design of the other aspects of the control, the javascript integration approach and the container. There are two independent decisions to be made: browser instance to render process choice and application to browser process process choice. It is tentatively decided that the there will be a separate render process for each top level (non-popup) instance of the browser. The application to browser process decision will have significant impact on the javascript integration related to application plugins. If multiple applications (developed by different teams) are allowed to run under a single instance of the browser process, an isolated environment for the execution of each application's plugins need to be setup. If on the other hand there will be a separate browser process for each application or a family of applications, the isolation requirement disappears as a misbehaving application plugin has no chance of affecting other applications.

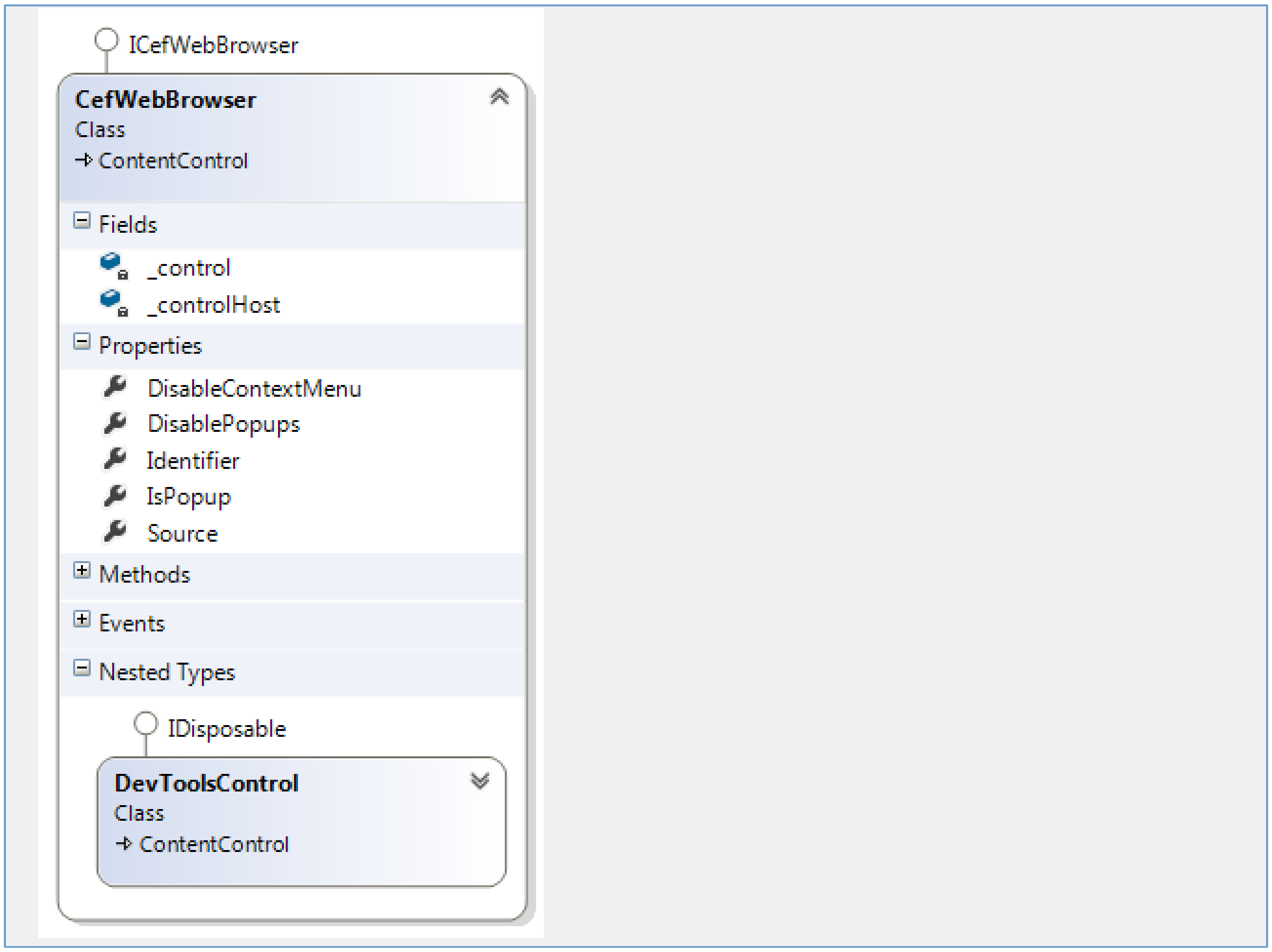
High level design



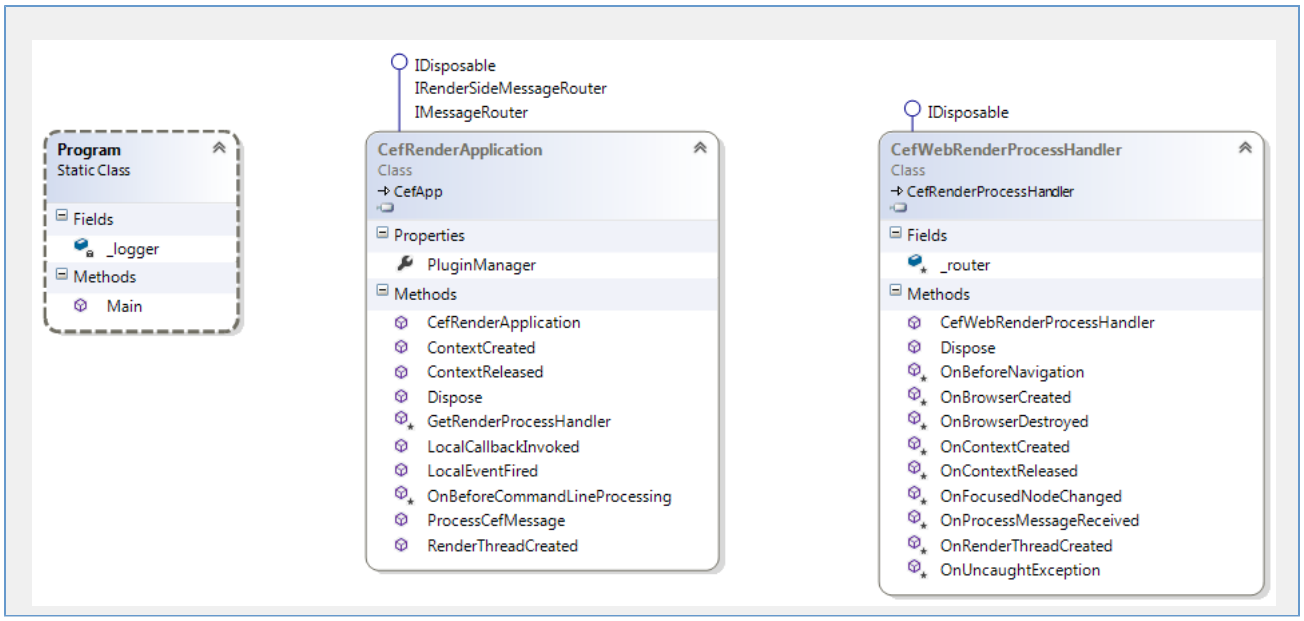
Details of significant classes in Windows Forms Control



Details of significant classes in WPF Control



Details of significant classes in the Render Process



Description of significant entities in the design

Details of the major aspects of controls behavior

CEF Initialization

The very first instance of the control loads CEF3 runtime and initializes CEF. A singleton CEF Browser Application instance is created at the time of CEF initialization. Control exposes a static event that gets fired just before CEF initialization. This event gives the containers a chance to set and/or modify the settings and also add javascript plugins to the Plugin Manager contained inside the CEF Browser Application.

CEF shutdown

Then control shuts down CEF when the very last top level instance closes.

Control lifetime management

Each instance of a control receives notifications from CEF separately regarding it's own lifetime management. The control also maintains a list of running instances (both top-level and popup). When a top level instance closes, it's corrsponding popup instances are closed automatically. When all top-level instances are closed, the control shuts down CEF.

Popup management

The control recognizes popup instances of itself. If the container indicated that Popups are allowed, the control fires an event to the container giving it an opprtunity to specify the settings for the new popup. Once the popup is created the control fires another event, giving the container the opportunity to create a custom frame window for the new popup. If the container does not create a custom frame window, the controls indicates to CEF to create a system provided frame window for the new popup. The control maintains the relationship between popups and their parents.

JavaScript dialogs

The control provides the default implementations for JavaScript dialogs like alerts, prompts etc. to address a major bug with chromium. The control first gives the container a chance to provide custom implementations for these dialogs. If the container fails to do so, the control uses the default implementations.

Context menu handling

The control provides a range of events to containers to handle context menus; both at page level and also at element level. If the containers do not handle context menus and the context menus are not suppressed, the default context menus are displayed.

Content loading

The control can be loaded with content either by setting the source property to a URL (normal url or a URI to a package) or by explicitly loading an application. The control, if it figures out that a packaged application is being loaded sets up the CEF resource handler that can handle requests from packages. It also sets up the default CORS settingsf or the application that allow access to other servers in the .gs.com domain.

The default url is "about:blank". The best time to bootstrap the content loading is just before the underlying CEF browser is created. The control exposes an event that gets fired just before the underlying browser is created allowing the container to specify the startup url and also the various settings for the browser.

Browser cache

The control allows a separate cache for each application.