Laboratorio di informatica – classe 4ª – data 5/4/2018

Web browser

Realizzare un internet browser (Windows Forms o WPF), che implementi le seguenti funzionalità:

- Navigazione di base:
  - a un determinato URL;
  - alla pagina precedente / successiva;
  - alla home page.
- Salvataggio della pagina corrente.
- Gestione segnalibri.
  - Inserimento/eliminazione/navigazione/salvataggio automatico.
- Gestione della configurazione dell'applicazione:
  - Consentire di stabilire la pagina iniziale.
  - Salvare/caricare la configurazione.

Segue il *mockup* di un'ipotetica UI.

Note sull'implementazione

Progettare la UI e la gestione dei dati in termini di "componenti". Alcuni suggerimenti:

- Una classe che gestisca i segnalibri, notificando quando un segnalibro viene aggiunto, eliminato, etc.
  - Questa, o un altra classe, dovrà provvedere a salvare/caricare i segnalibri su disco in una cartella ben definita del profilo utente.

- UI: una classe che gestisca la visualizzazione dei segnalibri e notifichi quando ne viene selezionato uno.
- Una classe che gestisca la configurazione.
  - Questa o un'altra classe dovrà provvedere a salvare/caricare la configurazione su disco in una cartella ben definita del profilo utente.

Componente di navigazione

Al centro dell'applicazione sta il componente di *browsing*. È possibile scegliere tra[1]:

1. *WebBrowser*: è un wrapper del motore di IExplorer; è già disponibile nelle applicazioni Windows Forms e WPF, e non richiede alcuna configurazione.

   (https://msdn.microsoft.com/en-us/library/system.windows.forms.webbrowser.aspx)

2. *CefSharp*: è basato sul progetto Chromium (a sua volta usato in Chrome). Deve essere installato e configurato.

   (https://cefsharp.github.io)

Il primo è di immediato utilizzo, ma è ormai obsoleto e dunque incompatibile con molti siti web. Il secondo è estremamente potente e continuamente aggiornato.

Progetto preconfigurato che usa CefSharp

CefSharp pone qualche problema di setup; inoltre, l'installazione mediante pacchetto Nuget pesa quasi 500Mbyte. Per questo scopo, ho creato un progetto di base, già configurato, che riduce il peso a circa 90Mbyte. (WebBrowserSharp)

(Per una soluzione avanzata al problema della scelta del componente, vedi il paragrafo "**Astrarre la UI dal componente di *browsing* usato"**).

Funzionalità di base

Segue un elenco delle funzioni di base fornite da entrambi i componenti:

Funzione	WebBrowser	CefSharp
Navigazione	`Navigate()`	`Load()`
Indietro	`GoBack()` `(CanGoBack())`	`Back()` `(CanGoBack())`
Avanti	`GoForward()` `(CanGoForward())`	`Forward()` `(CanGoForward())`
Pagina caricata (evento)	`DocumentCompleted` `Navigated`	`LoadingStateChanged` `AddressChanged`

(Nota bene: alcune operazioni possono essere asincrone.)

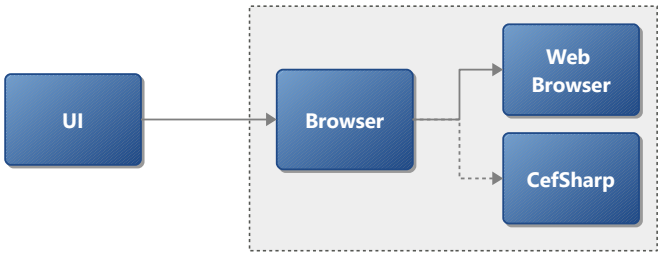
1 In realtà ne esistono altri.

Astrarre la UI dal componente di *browsing* usato

Poniamoci questo obiettivo: realizzare il programma senza che la UI debba dipendere da uno specifico componente di *browsing*. (D'ora in avanti: CB) Considerando che il *browsing* sta al centro dell'intera applicazione, appare un obiettivo impossibile da raggiungere, ma non è così.

Opzione 1: incapsulare il componente di browsing

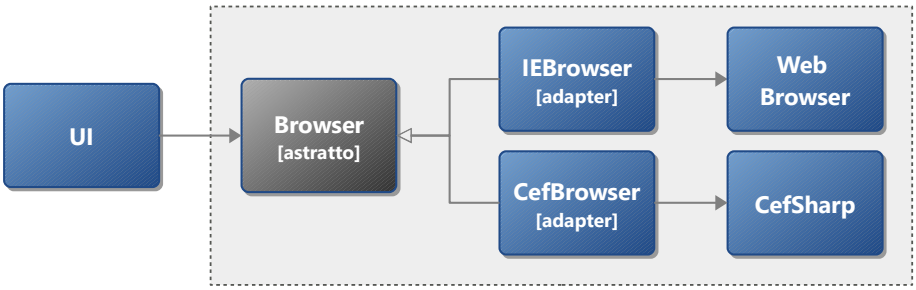
Si tratta di creare un "wrapper" intorno al CB, definendo un proprio componente di *browsing*. La classe definisce tutte le operazioni utilizzate dall'applicazione e le delega al CB effettivamente utilizzato.



Così facendo, se si decide di utilizzare un altro CB, sarà necessario modificare soltanto l'implementazione della classe `Browser`, e non l'intera UI.

Opzione 2: astrarre il componente browsing: Adapter Pattern

Un approccio più sofisticato prevede di disaccoppiare completamente la UI dal CB mediante un tipo astratto.



La UI dipende unicamente dal tipo astratto, il quale definisce tutte le operazioni di *browsing* necessarie. Questo viene implementato da uno o più "adattatori", e cioè classi che adattano (appunto) le funzioni richiese dalla UI a quelle fornite dai CB reali.

Si può iniziare creando l'adattatore per il componente *WebBrowser*; successivamente, si può implementare l'adattatore per il componente *CefSharp*, senza dover modificare l'applicazione se non il codice che stabilisca quale adattatore usare.