Autofill: tie key press handling to frames

vabr@chromium.org
Status: implemented in https://codereview.chromium.org/2762233004/
This document is publicly accessible. Last updated 6 April 2017.

Background

Autofill browser code displays suggestion pop-ups for form elements and handles key press events on them. Key handling is done by registering a handler with a RenderWidgetHost. A RenderWidgetHost can be accessed through RenderViewHost or a RenderFrameHost. With OOPIF present, not all frames have the same RenderWidgetHost, so accessing through RenderViewHost might not yield the right one. This resulted in https://crbug.com/688848.

The current autofill design ties a PopupControllerCommon to a WebContents instead of a particular RenderFrameHost. On the call stack up to PopupControllerCommon::RegisterKeyPressCallback, all the underlying stack frames have some association to the frame (mostly through the AutofillPopupDelegate, which is one per frame), but PopupControllerCommon only has WebContents and needs to derive the correct frame from it. This derivation is prone to errors. Unregistering key press handlers is similarly difficult and there is no guarantee that it happens if the frame is destroyed.

Goal

Tie registering and unregistering key press handlers to the correct frame and its lifetime.

Current code

- **PopupControllerCommon** is a class used by both AutofillPopupControllerImpl and PasswordGenerationPopupController to (un)register key press handlers. It keeps a pointer to WebContents and tries to deduce the correct frame to register the handlers with. Other than that PopupControllerCommon is only used to aggregate the popup-associated element's bounds, text direction and container view.
- **Registering key press handlers** is done through RenderWidgetHost, which is accessible through a RenderWidgetHostView, which is accessible through a RenderFrameHost.
- **AutofillPopupViewDelegate** subclasses involved in registering and unregistering the key press handlers are AutofillPopupControllerImpl and PasswordGenerationPopupControllerImpl

Proposal

Changes to Drivers

Because registering a key press handler is tied to a frame, it should be done through the frame-related ContentAutofillDriver, which should expose methods for registering and unregistering a RenderWidgetHost::KeyPressEventCallback. The methods are specific to the particular embedder of the autofill component (key handlers on other platforms like iOS might not make sense), so should not be included in the parent API AutofillDriver.

To avoid bloating of ContentAutofillDriver and also allow better unit-testability, a new class, KeyPressHandlerManager, will be created to provide the //content-independent logic (e.g., remembering the callback to register, dealing with a new callback replacing an old one, duplicate calls for registration, etc.):

```
class KeyPressHandlerManager {
 public:
  class Delegate {
   public:
    virtual void AddHandler(const KeyPressEventCallback& handler) = 0;
    virtual void RemoveHandler(const KeyPressEventCallback& handler) = 0;
  };
  void RegisterKeyPressHandler(const KeyPressEventCallback& handler);
  void RemoveKeyPressHandler();  // Unregisters previous handler.
};
```

The ContentAutofillDriver will implement the delegate (by forwarding to its frame's GetView()->GetRenderWidgetHost()) for KeyPressHandlerManager and aggregate an instance of KeyPressHandlerManager. Because drivers are naturally tied to a RenderFrameHost and its lifetime, the target frame for handler registration will always be clearly determined and non-null. Note that even in a valid RenderFrame the result of GetView() might be null, meaning that there is also no associated RenderWidgetHost and hence nothing to register the keypress callback with. This can happen in situations like a crashed frame or a frame lingering from past navigation. In those situations it is OK to silently fail registering the handler.

To help access the AutofillDriver from a PasswordManagerDriver, PasswordManagerDriver will expose a new getter, GetAutofillDriver(), for that. Inside the ContentPasswordManagerDriver implementation, this will be implemented as autofill::ContentAutofillDriver::GetForRenderFrameHost(render_frame_host_). Outside of that implementation the getter can be left NOTIMPLEMENTED(), but it is necessary to expose it through the content-independent API, because the access to the ContentAutofillDriver is needed inside the content-unaware AutofillPopupDelegate (see below).

Changes to AutofillPopupControllerImpl and PasswordGenerationPopupControllerImpl

PasswordGenerationPopupControllerImpl has a direct pointer to ContentPasswordManagerDriver. The ContentAutofillDriver is then easily obtained through downcasting the GetAutofillDriver() result, which is safe inside //chrome.

AutofillPopupControllerImpl is constructed inside ChromeAutofillClient::ShowAutofillPopup and passed an AutofillPopupDelegate on construction. The delegate will expose an AutofillDriver* getter, the result of which can be safely down-casted to Content*Driver, because AutofillPopupControllerImpl is in //chrome. The implementations of AutofillPopupDelegate are PasswordAutofillManager and AutofillExternalDelegate. The former has access to PasswordManagerDriver::GetAutofillDriver(). The latter has direct access to an AutofillDriver*.

The popup controllers should skip going through the PopupControllerCommon and directly call the driver to (un)register the handler.

Changes to PopupControllerCommon

PopupControllerCommon being bypassed for registering the handlers reduces to just a struct holding a few data members. It should be declared as such.