[Public] Plumbing prerender trigger data for NewTabPagePageLoadMetricsObserver

Author: robertlin@
Input: toyoshim@, sky@, arthursonzogni@, nhiroki@, alexmos@, ...

Last update: 2024-03-05 (Created: 2024-02-29)

Background

NewTabPagePageLoadMetricsObserver is introduced CL, and it is desired to carry the NavigationHandleUserData about whether this navigation is triggered by NewTabPage to record

the metrics. It is easy for prerendering navigation as the infrastructure already exists. The difficulty lies in the path for prerender activation and normal NewTabPage navigation.

Goals

The NTP navigation starts in MostVisitedHandler::OnMostVisitedTileNavigation through => WebContentsImpl::OpenURL => Browser::OpenURLFromTab. (Diagram shown above) The navigation handle used for attaching user data is available in Browser::OpenURLFromTab.

The goal is to attach NavigationHandleUserData to the navigation handle generated in Browser::OpenURLFromTab for NewTabPagePageLoadMetricsObserver. Introducing other functions like OpenURL may lead to a diverted behavior from the current OpenURL in the future, so handling it in the existing code path is considered.

To achieve this, there are two ways to do so, one being attach the user data directly in Browser::OpenURLFromTab (third block), the other being acquire the information of the navigation handle in MostVisitedHandler::OnMostVisitedTileNavigation (first block) and handle it in the same function. The first direction requires plumbing information through content/, and one way is to modify OpenURLParams. The second direction requires obtaining the information of navigation handles, and it requires adding storage in WebContents and WebContentsDelegate, or changing the return value of OpenURL.

Restrictions

According to content/public/README.md, Methods in the API should be there because either content is calling out to its embedder, or the embedder is calling to content. There shouldn't be any methods which are used to call from the embedder to the embedder.
And the use case here is for passing information from chrome/most_visted_handler.cc to chrome/browser.cc, so no extra variable definition should be introduced in content/ unless the newly introduced variable/API is used by chrome/.

Potential Options

Option 1 make WebContents::OpenURL return a struct containing the pointer to the Navigation Handle or adding a pointer to a pointer to NavigationHandle(to pass generated navigation handle from WebContentsDelegate::OpenURLFromTab as an output) to WebContents::OpenURL

Pros
Get the navigation handle directly from the function call
Cons
WebContents::OpenURL and `Browser::OpenURLFromTab are` virtual functions, changing them will introduce many changes in classes which inherit WebContents and WebContentsDelegate.

Option 2 Store (WebContents as Key, navigation handle) tuples from OpenURL in WebContents and WebContentsDelegates. Compare the return value of WebContents::OpenURL, and use the navigation handle if WebContents matches.

Details
This implementation records all (WebContents as Key, navigation handle) tuples from OpenURL if the navigation handle is not nullptr (from browser.cc), and the map will be cleared once it is matched with a query to save space.
Pros
Few changes in content/ and no base::SupportsUserData copy issue.
Cons
Extra storage.

Option 3 Use the same pattern of OpenURLParams inheriting base::SupportsUserData, and use CloneDataFrom(const SupportsUserData& other) in the copy cases.

Implementation details

This method is targeting the missing parts when inheriting base::SupportsUserData. In addition to appending data in OpenURLParams, it is required to make dtor virtual and add CloneDataFrom(const SupportsUserData& other) overridden copy constructor and operator.
Pros
Less changes in content/
Cons

- SupportsUserData has virtual dtor, but OpenURLParams doesn't. This will be unreliable.
- Involving copy will be more costly than other options. [cs]
- PreloadingDataImpl uses a similar pattern, but it disallows copy and assignment. Allowing this with OpenURLParams may introduce some risks.

Option 4: Adding extra parameters such as `std::optional<base::RepeatingCallback<void(NavigationHandle&)>> prerender_navigation_handle_callback`[cs] to WebContents::OpenURL

Implementation details
Basically another way of implementation, but the opposite direction of passing data to Option 1.
Pros
Straightforward implementation, and the correctness is guaranteed.
Cons
WebContents::OpenURL and `Browser::OpenURLFromTab` are virtual functions, changing them will introduce many changes in classes which inherit WebContents and WebContentsDelegate.

Options Attempted But Not Feasible

Option 1: Make OpenURLParams inherit base::SupportsUserData

Implementation details

Pros

- Still changes in content/, but relatively manageable.
- OpenURLParams is 1:1 for navigation, so the correctness is guaranteed.

Why this has concerns

- base::SupportsUserData has no copy operator nor copy constructor, but there are some existing OpenURLParams assignments. Not supporting this will lead to broken APIs.
- SupportsUserData has virtual dtor, but OpenURLParams doesn't. This will be unreliable.

Option 2: Get NavigationHandle data from Browser directly

Implementation details

1. Record last OpenURL/OpenURLFromTab information in Browser::OpenURLFromTab.
2. Get the navigation handle information by Browser::LastOpenURLInformation last_open_url_information = static_cast<Browser*>(web_contents_->GetDelegate()) ->GetLastOpenURLInformation(); directly in MostVisitedHandler.
3. Attach the user data to the navigation handle in MostVisitedHandler::OnMostVisitedTileNavigation.

Pros

- No content/ changes.

Why this doesn't work

- The assumption that Browser is the WebContentsDelegate is fragile and easily broken.
- Last Navigation Handle information may be overwritten.

Option 3: Store base::SupportsUserData in WebContents

1. Use WebContentsUserData to store information in WebContents
2. Extra the data in Browser

Pros

- No content/ changes.

Why this doesn't work

- WebContents and navigation is not 1-to-1 mapping, so the UserData may not be correct when it arrives at the point of retrieving user data.