

Manual History component: Requirements Analysis

Jason Eager

01 February 2000

1 Description of component

The Manual History component is a XPCOM component whose interface will be described by nsIManualHistory.

The main purpose of the Manual History component is to allow the skin (or chrome) to keep a selective and limited history of items that are frequently used or accessed by the user without needing to alter or parse through items recorded in the session history component. By creating a “stream” of history that is separate from the global history and the session history, the chrome can safely decide to not record navigation events in the Manual History component without worrying about affecting the internal operations of the browser or the users’ expectation that all pages visited will be recorded in some kind of history.

2 Detailed list of requirements

2.1 General/Design Philosophy Requirements

1. This component should not rely on and should not use the Global history component or the Session History component.
2. The component should expect to be completely controlled by the chrome and not by any other component.
3. The list of history items should be represented in string form, both to callers and internally. No layout or RDF data types should be used in the component.
4. The component should never directly call the navigation operation of the browser. The Chrome should retrieve the history item and it should call the navigate operation instead.

2.2 Data Requirements

1. The component needs to keep a finite dataset of urls added to it by the chrome.
2. The component also needs to record or keep the relative order of when these urls were last added to it by the chrome.
3. The number of items in the dataset is finite, and should be determined upon construction (or instation) of the component.
4. The state of the last remaining instance of the component on exit of the browser should be persistence between exiting the browser and restarting the browser.
5. The persistence of the above data should be related to the storage and retrieval of the user’s profile data, A unique dataset should exist for each profile.

2.3 Functional Requirements

1. When the component is constructed (or instantiated), it should load its dataset from the user's profile.
2. The component will be able to accept a item in string form through an add function and determine whether that item is contained in its dataset.
 - If the entered item is already contained within the component's dataset, it should move to the "top" of the most recently entered list.
 - If the item is not contained in the component's dataset, then it should be added to the dataset in the next empty place in the dataset, and moved to the "top" of the most recently entered list.
 - If the component's dataset is full, the entered item should replace the item that has not been entered for the longest time (i.e. the item at the "bottom" of the recently entered list). The newly placed item should be placed at the "top" of the recently entered list.
3. The component will be able to accept a url in string form through a remove function and determine whether that url is contained in its dataset.
 - If the entered item is located in the dataset, it should be removed from the dataset.
 - If the entered item is not located in the dataset, no action should be taken by the component..
4. The component will be able to accept a command to clear its dataset. Upon receiving the command, it will (of course) clear its dataset.
5. The component will accept a command to return a list of recently entered items in order of how they were entered. The caller will be able to specify how many items the component should return.
 - If the number of items requested is negative or zero, the component should return a null list.
 - If the number of items requested is greater then zero but less then the total number of items in the dataset, the list returned should contain that number of items in it, starting from the item at the "top" of the recently visited list downward.
 - If the number of items requested is equal to or more then the number of spaces in the dataset, the component should return all items in its dataset in order of the recently entered list.
6. Upon destruction of the component, the component should save its current dataset in the user's profile.

2.4 Performance Requirements

1. This component is not expected to have performance problems. However, the user should not be able to discern a notable pause when the component is performing any of its actions.
2. The design of the component (in terms of algorithms) will be based on the implicit assumption that the dataspace will not regularly exceed 500 items. If larger datasets are desired, then more attention should be paid to performance issues.

2.5 Other requirements

None at the present time.

3 Evaluation and comments

Comments about this document or about the Manual History component in general should be directed to

Jason Eager (e-mail address: jce2@po.cwru.edu)

This component is also listed in Bugzilla as enhancement/bug number 25735. .