Archived:Landmarks category selector dialog leaks memory (Known Issue)

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Description

The `TLmkItemIdDbCombiInfo` class provides methods for getting the ID of a landmark or category and the database handle to which the selected landmark or category belongs.

The client application executes methods to get the ID of a landmark and the database handle and then uses those for its internal purposes, such as for modifying the landmark attributes or for renaming the category. Note that the client takes the ownership of the returned database pointer and is responsible for deleting it.

Incorrect use of the Landmarks category selector dialog (`CLmkCategorySelectorDlg`) results in a memory leak whenever a category or categories are selected.

Solution

After selecting a category, the `TLmkItemIdDbCombiInfo` object contains a pointer to the landmarks database. The client is responsible for deleting this pointer once it is no longer needed.

```
#include <clmkcategoryselectordlg.h>  // link against lmkcommonui.lib
#include <tlmkitemiddbcombiinfo.h>

CLmkCategorySelectorDlg* dlg = CLmkCategorySelectorDlg::NewL(EFalse);

TLmkItemIdDbCombiInfo selectedItem;
TInt ret = dlg->ExecuteLD(selectedItem);

if( ret != 0 )
 {
 ...
 // Store the pointer to CPosLandmarkDatabase and
 // remember to delete it in the destructor.
 // The database pointer is the same for all landmarks
 // from the same database.
 ilDb = selectedItem.GetLmDb();
 }
```

See the **LocationLandmarksUIRefApp** SDK example application for a demonstration on using `CLmkCategorySelectorDlg`.

(C) Copyright Nokia 2014. All rights reserved.

 (C) Copyright Nokia 2014. All rights reserved.