
Subject: Doing my own thing with the ADD button on GridCtrl?

Posted by [jeremy_c](#) on Fri, 16 Oct 2009 18:36:23 GMT

[View Forum Message](#) <> [Reply to Message](#)

I would like to use common buttons on the GridCtrl's Toolbar such as Remove, Move item up, Move item down, etc... However the grid is only displaying a summary/small set of the actual data for the representing record. Therefore, when I add a new record, I do so via a Dialog box not direct entry into the GridCtrl.

How can I make the Appending and Inserting buttons not actually perform a GridCtrl append/insert but instead call my own function and let me call the dialog then later do a .Add()?

I tried immediately calling .CancelInsert() but that caused an Assertion failure in core\Value.h line 461.

Jeremy

Subject: Re: Doing my own thing with the ADD button on GridCtrl?

Posted by [jeremy_c](#) on Fri, 16 Oct 2009 18:46:54 GMT

[View Forum Message](#) <> [Reply to Message](#)

Oh... I was able to easily add to GridCtrl:

```
// .h
Callback WhenStartInsertRow;

// .cpp, void GridCtrl::DoAppend0(bool edit)
if (!WhenStartInsertRow.Empty())
{
    WhenStartInsertRow.Execute();
    return;
}
```

Which I can then use:

```
myGrid.WhenStartInsertRow = THISBACK(LoadEditDialog);
```

but... I was not sure if there was an existing way of doing this or not or if this addition was conforming to other uses of WhenStart* naming. I suppose that there should probably be two of these methods, one for Append and one for Insert. The Insert one passing a row index of where to insert or something along those lines.

What do you think?

Jeremy

Subject: Re: Doing my own thing with the ADD button on GridCtrl?

Posted by [unodgs](#) on Fri, 16 Oct 2009 19:00:57 GMT

[View Forum Message](#) <> [Reply to Message](#)

I usually do:

```
grid.StdAppend = THISBACK(ShowAddPlaylistMenu);
```

There is also StdInsert, StdRemove, StdEdit..

Subject: Re: Doing my own thing with the ADD button on GridCtrl?

Posted by [jeremy_c](#) on Fri, 16 Oct 2009 19:30:23 GMT

[View Forum Message](#) <> [Reply to Message](#)

Ah! Great. Guess I reinvented the wheel and probably not as good

Thanks,

Jeremy

Subject: Re: Doing my own thing with the ADD button on GridCtrl?

Posted by [sergeynikitin](#) on Sat, 17 Oct 2009 00:17:08 GMT

[View Forum Message](#) <> [Reply to Message](#)

I use the following scheme (I found these options, analyzing the examples, and in particular HomeBudget). Total 2 cases: 1 - edit in place. 2-editing from a separate window.

For edit in place I use next Callbacks:

WhenUpdateRecord - When editing the line after the last field has lost focus, to perform writing to disc.

(I can issue a command `grid.CancelUpdate ()` inside a CallBack to cancel the changes.)

WhenInsertRecord - When inserting/appending the line after the last field has lost focus, to perform writing to disc.

(I can issue a command `grid.CancelInsert ()` inside a CallBack to cancel the changes.)

WhenRemoveRecord - When deleting the line, to perform writing to disc.

(I can issue a command grid.CancelRemove () inside a CallBack to cancel the changes.)

For edit in separate window I use next Callbacks:

WhenStartEditing + IsNewRecord() filter - When editing the line.

WhenInsertRecord - When inserting/appending the line

Moreover, it is necessary in the first lines of procedure to issue CallBack CommitNewRow - to reset the flag a new line.

WhenRemoveRecord - When deleting the line, to perform prompting in separate window and writing to disc.

Example for edit in place:

Setting Callbacks:

```
NomencI.WhenInsertRow = THISBACK(InsertNomencI);  
NomencI.WhenUpdateRow = THISBACK(UpdateNomencI);  
NomencI.WhenRemoveRow = THISBACK(RemoveNomencI);
```

Actually the Callbacks:

```
void DictNomencI::InsertNomencI()  
{  
    sql * Insert(NOMENCL)  
        (NOM_NAME, NomencI(NOM_NAME));  
    NomencI.Refresh();  
    NomencI(NOM_ID) = sql.GetInsertedId();  
}  
void DictNomencI::UpdateNomencI()  
{  
    sql * SqlUpdate(NOMENCL)  
        (NOM_ID, NomencI(NOM_ID))  
        (NOM_NAME, NomencI(NOM_NAME))  
        .Where(NOM_ID == NomencI(NOM_ID));  
}  
void DictNomencI::RemoveNomencI()  
{  
    int i = (int)(NomencI(NOM_ID));  
    sql * Delete(NOMENCL).Where(NOM_ID == i);  
}
```

Example editing in separate window:

Setting Callbacks:

```
Company.WhenInsertRow = THISBACK(InsertCompany);  
Company.WhenRemoveRow = THISBACK(RemoveCompany);  
Company.WhenStartEdit = THISBACK(UpdateCompany);
```

Actually the Callbacks:

```
void DictCompany::InsertCompany()
{
    DictEditCompany dlg;
    Company.CommitNewRow();
    if(dlg.Run() != IDOK) {
        Company.CancelInsert();
        return;
    }
    sql * dlg.ctrls.Insert(COMpany);
    Company(COM_ID) = sql.GetInsertedId();
    Company(COM_NAME)=~dlg.tab1.COM_Name;
    dlg.SaveCompanyAddresses();
}

void DictCompany::UpdateCompany()
{
    if(Company.IsNewRow()) return;
    DictEditCompany dlg;
    dlg.Qptr = Company(COM_ID);
    sql * Select(dlg.ctrls).From(COMpany).Where(COM_ID == dlg.Qptr);
    if(!dlg.ctrls.Fetch(sql))
        return;
    dlg.LoadCompanyAddresses();
    if(dlg.Run() != IDOK) {
        Company.CancelEdit();
        return;
    }
    sql * dlg.ctrls.Update(COMpany).Where(COM_ID == dlg.Qptr);
    Company(COM_NAME)=~dlg.tab1.COM_Name;
    dlg.SaveCompanyAddresses();
}

void DictCompany::RemoveCompany()
{
    String p = t_("Delete company:")+String(Company(COM_NAME));
    if(PromptYesNo(p)){
        int i = (int)(Company(COM_ID));
        sql * Delete(COMpany).Where(COM_ID == i);
    } else
        Company.CancelRemove();
}
```

With this method GridCtrl himself work out almost all the logic, we can only issue a SQL command to put the data on the disk.

PS

This code works, but perhaps not the best. So I will sincerely grateful for even the smallest criticism and suggestions.
