Subject: Some addition proposals
Posted by crydev on Wed, 26 Apr 2017 10:30:25 GMT
View Forum Message <> Reply to Message

Hello,

During the development of my personal/hobby application CrySearch, I made several changes to the U++ framework. In this thread I would like to propose some of these to be added to the framework by default. I think it's a good idea, but let me know if you intentionally did not add it in the first place, or if you think it is not a good idea. :)

Firstly, my application contains some code that requires the title (text field) of a TabCtrl::Item. As such wasn't included at that moment, I added the following function to the TabCtrl::Item class. Good idea to add this to the framework?

```
const String&  GetText() const { return text; }
```

Secondly, I made a my own class derived from the ArrayCtrl, implementing functionality to set the background color of a row for each column at once. Similarly, I added a wrapper function for the Add(RowNum)Column functions that allow me to set the minimum width of the column. That avoids accidentally dragging a column such that it becomes unusable or invisible. Finally, I added functionality to retrieve a tuple of all indices that are currently visible to the user. This is useful because it allows you to update visible rows only. This last feature is built by Sender Ghost in this thread: http://www.ultimatepp.org/forums/index.php?t=msg&goto=39 908&#msg_39908. Finally, I added multi-row delete functionality with the DELETE key. :)

```
// Handles the DEL key for each array control in the application.
bool CrySearchArrayCtrl::Key(dword key, int count)
{
 // Check for DEL key presses.
 if (key == K_DELETE)
 {
  // Execute the removal routine. Since we make such intensive use of virtual rows, we
  // can't place responsibility at the control itself.
  Vector<int> selectedRows;
  const int rowCount = this->GetCount();
  for (int i = 0; i < rowCount; ++i)
  {
   if (this->IsSelected(i))
   {
    selectedRows << i;
   }
  }

  // If the removal routine was implemented, execute it. Otherwise, the DEL key does nothing.
```

```
  if (this->RemovalRoutine)
  {
   this->RemovalRoutine(selectedRows);
  }

  return true;
 }

 // Execute base key function of the ArrayCtrl class.
 return ArrayCtrl::Key(key, count);
}

// Returns a range of visible items in an array control.
Tuple2<int, int> CrySearchArrayCtrl::GetVisibleRange()
{
 int from = 0;
 int to = 0;
 const int last = this->GetCount() - 1;

 if (last >= 0)
 {
  const int cy = this->GetSize().cy - 1;
  const int sb = this->GetScroll();

  from = this->GetLineAt(sb),
  to = this->GetLineY(last) <= cy ? last : this->GetLineAt(sb + cy);
 }

 if (last < 0 || IsNull(to) || from > to)
 {
  from = to = -1;
 }

 return MakeTuple(from, to);
}

// Sets the same display structure for every cell in a row.
// The first parameter is the row and the second parameter is the display structure.
// This function requires protected access to the cellinfo vector.
void CrySearchArrayCtrl::SetRowDisplay(int i, const Display& d)
{
 const int rowCount = this->GetCount();
 const int colCount = this->GetColumnCount();
 if(i >= 0 && i < rowCount)
 {
  for (int j = 0 ; j < colCount; ++j)
  {
   this->cellinfo.At(i).At(j).Set(d);
```

```
  }
 }

 this->RefreshRow(i);
}

// Proxying method for adding a regular ArrayCtrl column. Sets a minimum width for a new
column.
ArrayCtrl::Column& CrySearchArrayCtrl::CryAddColumn(const char *text, int w)
{
 ArrayCtrl::Column& col = this->AddColumn(text, w);
 col.HeaderTab().Min(25);
 return col;
}

// Proxying method for adding a virtual data column. Sets a minimum width for a new column.
ArrayCtrl::Column& CrySearchArrayCtrl::CryAddRowNumColumn(const char *text, int w)
{
 ArrayCtrl::Column& col = this->AddRowNumColumn(text, w);
 col.HeaderTab().Min(25);
 return col;
}
```

The above code works fine, but every time I update U++, I have to move the 'cellInfo' variable in ArrayCtrl from private to protected. If some of this could be implemented in the framework itself, or maybe the cellInfo variable could be made protected by default, operations like above are easier to implement. :)

Thanks,

crydev
