"viewForTableColumn" vs "objectValueForTableColumn"

893

With xCode 9, in a NSTableView, when I use "objectValueForTableColumn" I must have columns with a "NSTableColumn" to see my data. On the other hand, when I use "viewForTableColumn" I can have the button from IB "NSTextFielCell". I do not understand why these two method does not work in the same way ?

In the same order of idea, I can programm the "return" key with "NSTextFielCell" but not with "NSTableColumn".

In the end, in my program, I add columns according to the number of fields which I want to handle. That works with "NSTableColumn" but I cannot use the return key. And with "NSTextFielCell" I can use the return key but I cannot show my data in the added columns. The function "NSTableCellView *cellView = [tableView makeViewWithIdentifier:tableColumn.identifier owner:self];" gives me an "nil" cell.

I think that I make something bad but what ?

Somebody would have an idea ?

AppKit

Answer this Question

Asked 3 years ago by ggBuguet

Add a Comment

Answers

— ***All*** NSTableViews use NSTableColumn for their columns.

— There are ***two*** ways of showing content in a table column:

The ***old*** way uses subclasses of NSCell (like NSTextFieldCell), called a "NSCell-based table view". The ***new*** way uses NSTableCellView, called a "view-based table view". You should always use a view-based table view, and never the old way. (The old way is supported so that existing code doesn't stop working.)

— In a view-based table view, you can still use "objectValueForTableColumn", but you don't have to. Your table cells (NSTableCellView) has an "objectValue" property, that must be made to refer to an object that supplies data to the controls inside the table cell (buttons, text fields, etc). There are ***three*** ways of doing that:

1. In your "viewForTableColumn" method, after you create the cell, you can simply set the property:

```
    NSTableCellView *cellView = [tableView
makeViewWithIdentifier:tableColumn.identifier owner:self];
    cellView.objectValue = …
```

2. In your data source, you can implement "objectValueForTableColumn", and return the desired objectValue.

3. You can use bindings.

Using #1 is probably the easiest way of doing this.

*>> The function "NSTableCellView *cellView = [tableView makeViewWithIdentifier:tableColumn.identifier owner:self];" gives me an "nil" cell.*

It's hard to say what might be wrong. The most likely reason is that your table column identifier does match the cell identifier in your prototype cell. If you're creating your own table columns, you must set their identifier to match the table cell(s) you are going to use in that column.

Posted 3 years ago by QuinceyMorris

Add a Comment

Thank you very much for your reply.

I can supply you more explanation.

In my "NSTableView" I want to show the components of the path of a file (5 or 6 components (fields)).

In IB I create two columns. I create the others by program:

```
nbcMax = 6;

for (NSInteger col = [[tableView tableColumns] count]; col < nbcMax; col++)

{

NSTableColumn *tc = [[NSTableColumn alloc] initWithIdentifier:[NSString
stringWithFormat:@"tc%ld", col]];

[[tc headerCell] setStringValue:[NSString stringWithFormat:@"%ld", col]];

[tableView addTableColumn:tc];

}
```

With "viewForTableColumn" only the first two columns, created with IB, are filled. The others exist but are empty :

```
- (NSView *)tableView:(NSTableView *)tableView viewForTableColumn:(NSTableColumn
*)tableColumn row:(NSInteger)row

{

NSTableCellView *cellView = [tableView makeViewWithIdentifier:tableColumn.identifier
owner:self];

NSArray *array = list[row];

NSInteger col = [[tableView tableColumns] indexOfObject:tableColumn];

if (col < [array count])

{

NSString *str = [array objectAtIndex:col];

NSLog(@"col : %ld - texte : %@", col, str);

if (col < [array count])

cellView.textField.stringValue = str;

}

return cellView;

}
```

If I create all columns (6) in IB, it works very well but I do not know the number of columns which I shall need.

If you have an idea ... Thanks in advance.

Posted 3 years ago by ggBuguet

Add a Comment

You can't do that. A table view is ***completely*** view-based or ***completely*** NSCell-based, and cannot be a mixture of the two. If you implement the "tableView:viewForTableColumn:row:" delegate method, the table view is view-based.

>> If I create all columns (6) in IB, it works very well but I do not know the number of columns which I shall need.

Create a table with 6 columns, then programmatically remove the columns you don't need. This is much easier than creating a table with 2 columns and adding more.

Posted 3 years ago by QuinceyMorris

Add a Comment

Thank you very much.

I am going to follow your recommendation but I do not understand why it works very well with 'tableView:objectValueForTableColumn'.

Posted 3 years ago by ggBuguet

Add a Comment

This site contains user submitted content, comments and opinions and is for informational purposes only. Apple disclaims any and all liability for the acts, omissions and conduct of any third parties in connection with or related to your use of the site. All postings and use of the content on this site are subject to the Apple Developer Forums Participation Agreement.

Copyright © 2022 Apple Inc. All rights reserved. Terms of Use | Privacy Policy | License Agreements