How would you implement an UITableView that detects horizontal swipes in order to allow paging?

Asked 16 years ago Modified 15 years, 3 months ago Viewed 9k times

6

I'd like to implement a UITableView which displays say 20 rows at a time. But given the fact that I might actually have say 120 items to present, I'd like to handle this using some sort of paging:

Populate table with first 20 items. When the user does a right-to-left swipe, reload the UITableView with the next 20 items. A left-to-right swipe would show the previous 20 items.

I think subclassing UITableView and handle UITouches there is the way to go, but I'm not sure how to handle the touches correctly in order not to interfer with the standard event handling.

Eg. if the user just touches (clicks) a row, I want the row to hightlight (selected) and then switch to a detail view presenting data corresponding to the selected row. But if the user does a horizontal swipe, I don't want the row below the swipe to be highlighted (and therefore selected), instead, i want to trigger my loadprevious page / loadnext page handling, which wil reload the table with

the according 20 previous or next items, depending on the swipe direction.

Any suggestions?

iphone cocoa-touch

Share

Improve this question

Follow

asked Dec 5, 2008 at 2:41

cardinal

7 Answers

Sorted by: Highest score (default)

9

A horizontal swipe on a table row is already a standard UI behavior that causes that row to be deleted. Don't go changing standard UI paradigms -- it confuses users and makes them dislike your app.

Share Improve this answer

Follow

answered Dec 5, 2008 at 3:58

August

12.2k 3 31 30

In the "Byline" app you can swipe items to mark them as read or unread, and I like that. – Chris Lundie Feb 16, 2009 at 17:56

How about using UITableView's section index as an alternative?

3

Override the following:

```
- (NSArray *)sectionIndexTitlesForTableView:(UITableVi

- (NSInteger)tableView:(UITableView *)tableView sectio
(NSString *)title atIndex:(NSInteger)index;
```

This will give you the index bar on the right side of the tableview which lets users jump quickly to the bottom of the table (See the iPod app for a reference if you don't know what I mean.) You don't have to include every section in the index either so you can keep from overloading it with 100's of sections.

I use the degree symbol for my index, since my table sections aren't alphabetic.

If the backing datastructure is still one big list you would need to break it up into sections. (You don't have to have a header show for sections, so it can look identical to the existing table)

As for departing from the standard UI behavior... occasionally when trying to issue an app update, you can get a cantankerous reviewer which will reject your update for a ui "issue" which was present in a previous version of the app. Just a warning, as when you're trying to get an important fix out, it's the most aggravating thing in the world. *shakes fist at Apple*

As for performance, I've had non-chunky scroll performance on tables up to 900 items. Things to watch

for if it gets chunky... 1) Ensure you using the Reuse Identifier correctly when dequeing cells. 2) Custom table cells with complex controls. (If you have to add your own sub views to the table cells, UILabels are fastest, with no transparency) 3) Avoid table cells of non-standard height (a.k.a. not 40). (an acknowledged apple performance bug, though I haven't validated this is really an issue.)

Share Improve this answer

Follow

answered Mar 30, 2009 at 20:06

Bingosabi

61 ● 1 ● 1

3

The main question to ask here is “why”: why do you think the user would want to go to item #527? How is he expected to know the item he wants is on page 6? On page 17?

You correctly point that scrolling through 1000 items is weird, but paging through 1000 items is equally weird. You probably want to improve filtering or sorting capabilities of your application. Scrolling through thousands of items is insane, unless you expect the user to read/scan each one, in which case having a linear list is not a problem.

UITableView has a section index on the right, which might solve your problem depending on the context. It certainly solves the problem for the contact list: even if you have 2000 contacts, you only need to scroll through about 100 of them once you pick the starting letter. (The

performance is not a problem, because UITableView never creates or asks for more rows than what is visible on the screen.)

Finally, I've used one app that did a similar paging + table view and I hated the experience. It feels completely non-iPhonish.

Share Improve this answer

Follow

answered May 7, 2009 at 22:04

Andrey Tarantsov

9,063 ● 7 ● 56 ● 59

2

This is quite a non-standard UI paradigm. Surely you'd scroll up/down over the 120 items, just like (say) the Contacts app. You're making life more complicated for yourself - and you run a serious risk of this app being rejected by Apple because of the odd UI.

The actual method by which you'd implement this is described in http://forums.macrumors.com/showthread.php?t=532573

I've read the code there, and it looks sensible, though it occurs to me that you may also need to handle touchUpInside, and not call [super touchUpInside] in order to avoid selecting a cell on a horizonal swipe.

Share Improve this answer

Follow

edited Dec 5, 2008 at 3:30

answered Dec 5, 2008 at 3:18

Airsource Ltd

32.6k ● 13 ● 74 ● 76

1

Just call a touch cancel method there check the swipe is occured ir not with touch points.Then do your functionality

Share Improve this answer

Follow

answered Feb 16, 2009 at 15:36

narendar Ellandula

1

If I really wanted to do this I would subclass UITableView and add multiple copies of it to a UIScrollView. Untangling the events would be dreadful and I don't think the UI would be worth the effort.

Share Improve this answer

Follow

edited Mar 7, 2009 at 19:02

answered Feb 16, 2009 at 23:07

Rog

17.2k ● 9 ● 51 ● 73

1

I actually wrote the code mentioned by Airsource Ltd at http://forums.macrumors.com/showthread.php?t=532573, so pointing me to my own code wasn't much help, thx



anyway, my fault, but hey, who doesn't use different screennames on different forums? ;-)

Anyway, it was approved after submission to the appstore, and IMHO it is very useful when dealing with long lists. Paging is something commonly used on the web, and I wouldn't like to force a user to scroll down/up a list with 500+ (did I say 500+? what about 1000+?) rows.

I agree that a horizontal swipe on a list should/will enable the user to delete/edit a row as a default behaviour. But isn't that context-specific?

If I as a user am the "owner" of the data, I would expect to be able to delete rows. But would I expect to be able to delete rows in a list of apps in the AppStore, or a list of books in an Amazon app?

A horizontal swipe on a list is just a gesture, depending on the context, it should do what the user would expect.

Given the scenario of a list with 500+ rows, what would you do? Show the first 25 rows with a "Show me more"-Footer? -> then you'd have 50 Rows? and so on? in the end, you'd have 527 rows? Unless you go hardcore-drawing you cell, the table will draw choppy too on top of it. But even if scrolling smoothly, this wouldn't resolve the problem of scrolling up/down 2000 miles of pixels.

Or put a Header/Footer with "Previous Page / Next Page" - Buttons, thus only showing 25 rows at a time? You'd

then force the user to scroll to the top or bottom to switch to the previous or next page....

I'm open for any suggestions. It appears just as a natural gesture to me to swipe left or right on something that has a "previous/next" something.

Share Improve this answer

Follow

answered Mar 24, 2009 at 0:33

codeclash

2,063 ● 19 ● 17