EBook Gratuito

APPRENDIMENTO uitableview

Free unaffiliated eBook created from
Stack Overflow contributors.

#uitableview

Sommario

Di

Capitolo 1: Iniziare con uitableview

Osservazioni

Questa sezione fornisce una panoramica di cos'è uitableview e del motivo per cui uno sviluppatore potrebbe volerlo utilizzare.

Dovrebbe anche menzionare tutti i soggetti di grandi dimensioni all'interno di uitableview e collegarsi agli argomenti correlati. Poiché la documentazione di uitableview è nuova, potrebbe essere necessario creare versioni iniziali di tali argomenti correlati.

Examples

UITableView in dettaglio

Cos'è UITableView?

`UITableView` è un oggetto di interfaccia utente utilizzato più di frequente che presenta i dati in un **elenco scorrevole** di più righe in una singola colonna che può anche essere diviso in sezioni. Consente solo lo scorrimento verticale ed è una sottoclasse di `UIScrollView` .

Perché usiamo UITableView?

Possiamo usare `UITableView` per presentare un **elenco di opzioni** che possono essere selezionate, per navigare attraverso **dati strutturati gerarchicamente** , presentare un **elenco indicizzato di elementi** , per visualizzare informazioni dettagliate e controlli in raggruppamenti visivamente distinti facendo uso di **sezioni** .

Possiamo vedere l'uso di `UITableView` nei nostri contatti, mailing list, ecc. Non è solo usato per presentare i dati testuali, ma anche le immagini e i testi possono essere elencati come nell'app YouTube.

Istruzioni dettagliate su come installare o installare UITableView usando Story Board.

1. Crea un progetto semplice per l'applicazione Vista singola.
2. Nella libreria degli oggetti, seleziona l'oggetto "Vista tabella" e trascinalo nella vista del tuo controller di visualizzazione. Semplicemente eseguendo il progetto vedrai una pagina vuota con le linee.
3. Ora se vuoi semplicemente una vista scorrevole con i contenuti, trascina un `UIView` in `UITableView` , regola le sue dimensioni e trascina il resto degli UIElements in quella vista secondo i requisiti. Ma se vuoi presentare un elenco di formato simile, usiamo `UITableViewCell` .
4. La classe `UITableViewCell` definisce gli attributi e il comportamento delle celle visualizzate negli oggetti UITableView. Questa classe include proprietà e metodi per impostare e gestire il contenuto e lo sfondo delle celle (inclusi testo, immagini e viste personalizzate), gestire la selezione e lo stato delle evidenziazioni delle celle, gestire le viste accessorie e avviare la

modifica dei contenuti delle celle.

5. La parte migliore dell'uso di `UITableViewCell` è la riusabilità. Lo scopo di dequeueReusableCellWithIdentifier è quello di utilizzare meno memoria. Ad esempio, se si dispone di un elenco di 1000 voci e solo 10 voci sono visibili alla volta, solo le celle visibili vengono allocate in memoria, il resto viene riutilizzato come quando l'utente scorre l'elenco. Tutto ciò che devi fare è trascinare un `UITableViewCell` e rilasciarlo su tableView. Quindi fai clic su cell-> Vai a attributo inspector-> Imposta lo stile tableView su custom e identificatore su qualcosa di unico che vorresti dire myCell.
6. Ora dobbiamo conformarci all'origine dati in modo che l'oggetto dia dati a un altro oggetto. Ad esempio, il protocollo `UITableViewDataSource` ha metodi come cellForRowAtIndexPath e `numberOfRowsInSection` dettano ciò che dovrebbe essere visualizzato nella tabella. Mentre l'oggetto del tipo delegato risponde alle azioni eseguite da un altro oggetto. Ad esempio, il protocollo `UITableViewDelegate` ha metodi come `didSelectRowAtIndexPath` per l'esecuzione di azioni su un utente che seleziona una particolare riga in una tabella. 7. Quando si conforma a datasource è necessario implementare il suo metodo richiesto ie

```
 - (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
        // Here you need to give the total number of items you want to list. For example if
you want list of 2 numbers, then:

           return 2;

 }
```

e

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath {

//Here you need to dequeue the reusable cell which we discussed in point 5. Then you can
modify your cell here according to you and customize it here as per your requirement. Here the
call comes for numberOfRows number of times.

static NSString *cellIdentifier = @"cellID";

UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:
cellIdentifier];
if (cell == nil) {
    cell = [[UITableViewCell alloc]initWithStyle:
    UITableViewCellStyleDefault reuseIdentifier:cellIdentifier];
}
if(indexPath.row){
    [cell.textLabel setText:"1"];
}else{
    [cell.textLabel setText:"2"];
}

return cell;
}
```

7. Inoltre su NSIndexPath: - La classe NSIndexPath rappresenta il percorso di un nodo specifico in un albero di raccolte di array nidificate. Questo percorso è noto come un percorso dell'indice. I suoi oggetti sono sempre di lunghezza 2. Sono utilizzati ad esempio

per indicizzare una cella di visualizzazione tabella. Il primo indice in un oggetto NSIndexPath è chiamato la sezione, il secondo è la riga. Un oggetto percorso indice con sezione 0 e riga 0 indica la prima riga nella prima sezione. Utilizzare `[NSIndexPath indexPathForRow:inSection:]` per creare rapidamente un percorso dell'indice.

Metodi di delega e origine dati:

Ogni vista tabella deve avere un delegato e un'origine dati.

Metodi delegati

Nessuno dei metodi delegati è effettivamente richiesto, tuttavia è necessario implementare tableView: didSelectRowAtIndexPath: per gestire i tocchi su una cella della tabella:

E altri metodi sono ...

```
// Display customization

- (void)tableView:(UITableView *)tableView willDisplayCell:(UITableViewCell *)cell
forRowAtIndexPath:(NSIndexPath *)indexPath;

// Variable height support

// If these methods are implemented, the above -tableView:heightForXXX calls will be deferred
until views are ready to be displayed, so more expensive logic can be placed there.

- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath
*)indexPath;
- (CGFloat)tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section;
- (CGFloat)tableView:(UITableView *)tableView heightForFooterInSection:(NSInteger)section;

// Section header & footer information. Views are preferred over title should you decide to
provide both

- (UIView *)tableView:(UITableView *)tableView viewForHeaderInSection:(NSInteger)section;   //
custom view for header. will be adjusted to default or specified header height
- (UIView *)tableView:(UITableView *)tableView viewForFooterInSection:(NSInteger)section;   //
custom view for footer. will be adjusted to default or specified footer height

// Accessories (disclosures).

- (void)tableView:(UITableView *)tableView
accessoryButtonTappedForRowWithIndexPath:(NSIndexPath *)indexPath;

// Selection

// Called before the user changes the selection. Return a new indexPath, or nil, to change the
proposed selection.
- (NSIndexPath *)tableView:(UITableView *)tableView willSelectRowAtIndexPath:(NSIndexPath
*)indexPath;
// Called after the user changes the selection.
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath;


// Editing
```

```
// Allows customization of the editingStyle for a particular cell located at 'indexPath'. If
not implemented, all editable cells will have UITableViewCellEditingStyleDelete set for them
when the table has editing property set to YES.
- (UITableViewCellEditingStyle)tableView:(UITableView *)tableView
editingStyleForRowAtIndexPath:(NSIndexPath *)indexPath;

// Controls whether the background is indented while editing.  If not implemented, the default
is YES.  This is unrelated to the indentation level below.  This method only applies to
grouped style table views.
- (BOOL)tableView:(UITableView *)tableView shouldIndentWhileEditingRowAtIndexPath:(NSIndexPath
*)indexPath;



// Indentation

- (NSInteger)tableView:(UITableView *)tableView indentationLevelForRowAtIndexPath:(NSIndexPath
*)indexPath; // return 'depth' of row for hierarchies
```

Metodi richiesti per i dati:

I seguenti metodi sono richiesti dall'origine dati: tableView: numberOfRowsInSection: e tableView: cellForRowAtIndexPath :. Se la tabella è una tabella raggruppata, è necessario implementare numberOfSectionsInTableView :.

Metodi richiesti : -

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section;

// Row display.

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath;
```

Metodi opzionali : -

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView;              // Default is
1 if not implemented

- (NSString *)tableView:(UITableView *)tableView titleForHeaderInSection:(NSInteger)section;
// fixed font style. use custom view (UILabel) if you want something different
- (NSString *)tableView:(UITableView *)tableView titleForFooterInSection:(NSInteger)section;

// Editing

// Individual rows can opt out of having the -editing property set for them. If not
implemented, all rows are assumed to be editable.
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath;

// Moving/reordering

// Allows the reorder accessory view to optionally be shown for a particular row. By default,
the reorder control will be shown only if the datasource implements -
tableView:moveRowAtIndexPath:toIndexPath:
- (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:(NSIndexPath *)indexPath;
```

```
// Index

- (NSArray<NSString *> *)sectionIndexTitlesForTableView:(UITableView *)tableView;
// return list of section titles to display in section index view (e.g. "ABCD...Z#")
- (NSInteger)tableView:(UITableView *)tableView sectionForSectionIndexTitle:(NSString *)title
atIndex:(NSInteger)index;  // tell table which section corresponds to section title/index
(e.g. "B",1))

// Data manipulation - insert and delete support

// After a row has the minus or plus button invoked (based on the UITableViewCellEditingStyle
for the cell), the dataSource must commit the change
// Not called for edit actions using UITableViewRowAction - the action's handler will be
invoked instead
- (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath
*)indexPath;

// Data manipulation - reorder / moving support

- (void)tableView:(UITableView *)tableView moveRowAtIndexPath:(NSIndexPath *)sourceIndexPath
toIndexPath:(NSIndexPath *)destinationIndexPath;
```

Anatomia di una cella di tabella

L'UITableViewCell predefinito ha diverse visualizzazioni di dati e visualizzazioni secondarie standard.

- **cell.textLabel** : l'UILabel per la cella
- **cell.detailTextLabel** - una UILabel più piccola che appare sotto l'etichetta di testo
- **cell.imageView** - a UIImageView che il lato sinistro della cella

La vista accessorio opzionale può contenere una delle seguenti icone. L'accessorio **defaultType** è **UITableViewCellAccessoryNone** .

Leggi Iniziare con uitableview online: https://riptutorial.com/it/uitableview/topic/6178/iniziare-con-uitableview

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con uitableview	Community, Ishika, Rahul