eBook Gratuit

APPRENEZ uitableview

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#uitableview

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: uitableview

It is an unofficial and free uitableview ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official uitableview.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec uitableview

Remarques

Cette section fournit une vue d'ensemble de ce qu'est une vue et de la raison pour laquelle un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les sujets importants dans une vue accessible, et établir un lien avec les sujets connexes. La documentation de uitableview étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Examples

UITableView en détail

Qu'est-ce que UITableView?

`UITableView` est un objet d'interface utilisateur le plus fréquemment utilisé qui présente des données dans une **liste déroulante** de plusieurs lignes dans une seule colonne pouvant également être divisée en sections. Il permet uniquement le défilement vertical et constitue une sous-classe de `UIScrollView` .

Pourquoi utilisons-nous UITableView?

Nous pouvons utiliser `UITableView` pour présenter une **liste d'options** pouvant être sélectionnées, pour naviguer dans **des données structurées hiérarchiquement** , présenter une **liste indexée d'éléments** , afficher des informations détaillées et des contrôles dans des groupes visuellement distincts utilisant des **sections** .

Nous pouvons voir l'utilisation de `UITableView` dans nos contacts, listes de diffusion, etc. Ce n'est pas seulement utilisé pour présenter des données textuelles, mais aussi des images et des textes peuvent être listés comme dans l'application YouTube.

Instructions détaillées sur la configuration ou l'installation d'ITUableView à l'aide de Story Board.

1. Créez un projet simple pour l'application Single View.
2. Dans la bibliothèque d'objets, sélectionnez l'objet «Table View» et faites-le glisser dans la vue de votre contrôleur de vue. En exécutant simplement le projet, vous verrez une page vierge avec des lignes.
3. Maintenant, si vous voulez simplement une vue défilante avec du contenu, faites glisser un `UIView` dans `UITableView` , ajustez sa taille et faites glisser le reste des UIElements dans cette vue, conformément aux exigences. Mais si vous souhaitez présenter une liste de format similaire, nous utilisons `UITableViewCell` .
4. La classe `UITableViewCell` définit les attributs et le comportement des cellules qui apparaissent dans les objets UITableView. Cette classe comprend des propriétés et des

méthodes pour définir et gérer le contenu et l'arrière-plan des cellules (texte, images et vues personnalisées), gérer la sélection de cellules et l'état de mise en évidence, gérer les vues d'accessoires et éditer le contenu de la cellule.

5. La meilleure partie de l'utilisation de `UITableViewCell` est la réutilisation. Le but de deReueReusableCellWithIdentifier est d'utiliser moins de mémoire. Par exemple, si vous avez une liste de 1000 entrées et que seules 10 entrées sont visibles à la fois, seules les cellules visibles sont allouées, les autres sont réutilisées. Tout ce que vous avez à faire est de faire glisser un objet `UITableViewCell` et de passer à tableView. Cliquez ensuite sur cell-> Go pour attribuer l'inspecteur-> Définir le style tableView sur custom et l'identifiant à tout ce que vous voudriez dire dire myCell.
6. Maintenant, nous devons nous conformer à la source de données pour que l'objet donne des données à un autre objet. Par exemple, le protocole `UITableViewDataSource` a des méthodes telles que cellForRowAtIndexPath et `numberOfRowsInSection` qui `numberOfRowsInSection` ce qui doit être affiché dans la table. Considérant que l'objet de type délégué répond aux actions prises par un autre objet. Par exemple, le protocole `UITableViewDelegate` a des méthodes telles que `didSelectRowAtIndexPath` pour effectuer des actions `didSelectRowAtIndexPath` utilisateur sélectionne une ligne particulière dans une table.
7. Lorsque vous vous conformez à la source de données, vous devez implémenter la méthode requise, à savoir

```
 - (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;

 }
```

et

```
- (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. A propos de NSIndexPath: - La classe NSIndexPath représente le chemin d'accès à un nœud spécifique dans une arborescence de collections de tableaux imbriqués. Ce chemin est appelé chemin d'index. Ses objets sont toujours de longueur 2. Ils sont utilisés par exemple pour indexer une cellule de vue tableau. Le premier index d'un objet NSIndexPath s'appelle la section, le second est la ligne. Un objet de chemin d'index avec la section 0 et la ligne 0 indique la première ligne de la première section. Utilisez `[NSIndexPath indexPathForRow:inSection:]` pour créer rapidement un chemin d'index.

Méthodes de délégué et de source de données:

Chaque vue de table doit avoir un délégué et une source de données.

Méthodes de délégué

Aucune des méthodes de délégation n'est en réalité requise, mais vous devrez implémenter tableView: didSelectRowAtIndexPath: pour gérer les contacts sur une cellule de tableau:

Et d'autres méthodes sont ...

```
// 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
```

Méthodes requises pour la source de données:

Les méthodes suivantes sont requises à partir de la source de données: tableView: numberOfRowsInSection: et tableView: cellForRowAtIndexPath :. Si votre table est une table groupée, vous devez également implémenter numberOfSectionsInTableView :.

Méthodes requises : -

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

// Row display.

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

Méthodes optionnelles : -

```
- (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;
```

Anatomie d'une cellule de table

UITableViewCell par défaut a plusieurs vues et sous-vues de données standard.

- **cell.textLabel** - le UILabel pour la cellule
- **cell.detailTextLabel** - un UILabel plus petit qui apparaît sous l'étiquette de texte
- **cell.imageView** - un UIImageView que le côté gauche de la cellule

La vue des accessoires en option peut contenir l'une des icônes suivantes. Le type d'accessoire par défaut est **UITableViewCellAccessoryNone** .

Lire Commencer avec uitableview en ligne:
https://riptutorial.com/fr/uitableview/topic/6178/commencer-avec-uitableview

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec uitableview	Community, Ishika, Rahul