eBook Gratuit

APPRENEZ autolayout

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

#autolayout

Table des matières

À propos

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

It is an unofficial and free autolayout 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 autolayout.

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: Démarrer avec autolayout

Remarques

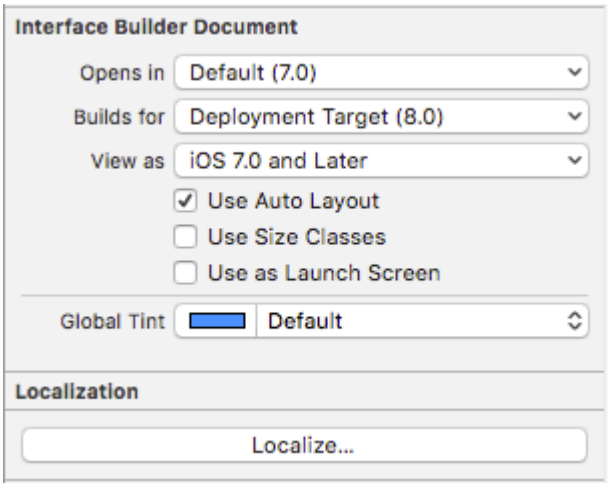
La mise en page automatique calcule dynamiquement la taille et la position de toutes les vues de votre hiérarchie de vues, en fonction des contraintes imposées à ces vues - par Apple. La mise en page automatique peut également être comprise en la décrivant comme un ensemble de règles que vous utilisez pour chacune des vues de votre hiérarchie de vues. Cette règle définit la façon dont l'alignement et la réduction ou l'extension de la vue seront effectués en fonction des différentes tailles d'écran.

Nous devrions utiliser Auto Layout dans tous nos projets car cela nous aide à gérer la conception des écrans pour tous les périphériques disponibles que nous ciblons. Pour cela, il suffit de comprendre ce que c'est et comment il fonctionne, et après avoir compris comment cela fonctionne, il sera amusant de concevoir une disposition dynamique pour plusieurs périphériques.

Examples

Installation ou configuration

Pour utiliser la mise en page automatique, vous devez activer un indicateur booléen dans le storyboard. Son montré dans l'image ci-dessous. Après cela, nous pouvons utiliser la mise en page automatique dans notre storyboard. Il y a une autre option si vous voulez utiliser des classes de taille ou non. La classe de taille est également une option utile dans la mise en page automatique qui nous aide à concevoir différentes tailles d'écran de périphérique différemment avec des éléments identiques ou différents. L'image pour la configuration va ici. C'est à partir du panneau d'inspection de fichiers de l'éditeur de propriétés dans Xcode.



Ajout d'un UIImageView au centre de l'écran

Si vous souhaitez ajouter un UIImageView au centre de l'écran avec une largeur et une hauteur de 100 pixels, vous devez définir la contrainte center x et la contrainte center y sur la contrainte

UIImageView et width, height sur UIIMageView. Voici le code. Il y a moyen de le faire dans le storyboard mais j'ai écrit en code parce que c'est compréhensible pour cette situation dans la documentation.

```
UIImageView *imageView = [[UIImageView alloc] initWithFrame:CGRectMake(0, 0, 100.0, 100.0);
imageView.self.translatesAutoresizingMaskIntoConstraints = NO;

[imageView addConstraint:[NSLayoutConstraint constraintWithItem:imageView
attribute:NSLayoutAttributeWidth relatedBy:NSLayoutRelationEqual toItem:nil
attribute:NSLayoutAttributeNotAnAttribute multiplier:1.0 constant:100.0]];
[imageView addConstraint:[NSLayoutConstraint constraintWithItem:imageView
attribute:NSLayoutAttributeHeight relatedBy:NSLayoutRelationEqual toItem:nil
attribute:NSLayoutAttributeNotAnAttribute multiplier:1.0 constant:100.0]];
[superview addConstraint:[NSLayoutConstraint constraintWithItem:imageView
attribute:NSLayoutAttributeCenterX relatedBy:NSLayoutRelationEqual toItem:superview
attribute:NSLayoutAttributeCenterX multiplier:1.0 constant:0]];
[superview addConstraint:[NSLayoutConstraint constraintWithItem:imageView
attribute:NSLayoutAttributeCenterX relatedBy:NSLayoutRelationEqual toItem:superview
attribute:NSLayoutAttributeCenterX multiplier:1.0 constant:0]];
```

Utilisation du langage de format visuel

Il existe un moyen de simplifier la définition de autolayout pour les vues à l'aide de VFL. Cela peut sembler difficile au début, mais c'est en fait très facile à utiliser. Quelques définitions d'abord:

- `|` représente superview
- `H:` ou `V:` représente l'orientation actuelle - horizontale ou verticale
- les noms de vue doivent être placés entre crochets
- la hauteur et la largeur de la vue doivent être placées entre parenthèses
- les marges sont spécifiées entre les vues et entourées par des tirets
- la priorité pour une marge ou une taille de vue peut être spécifiée avec @

Exemples de syntaxe de format visuel

1. `someView` est attaché aux bords gauche et droit de sa superview sans marges:

   ```
   H:|[someView]|
   ```

2. `someView` est attaché au sommet avec une marge de 10 points et a une hauteur égale à `value` :

   ```
   V:|-(10)-[someView(value)]
   ```

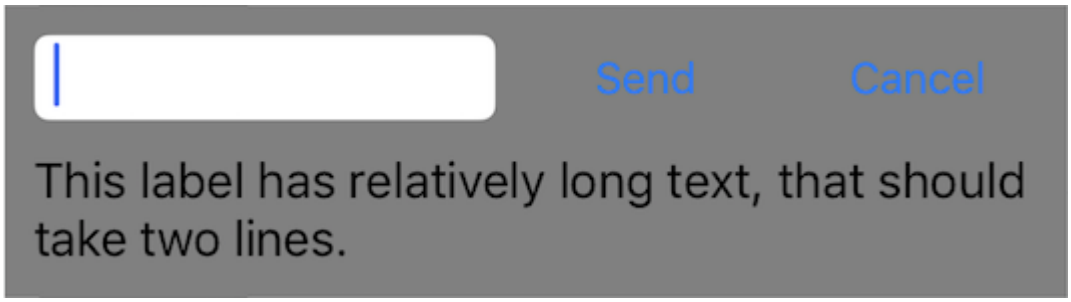
3. `someView1` et `someView2` ont deux marges définies entre eux - `value1` avec une priorité de 900 et `value2` avec une priorité de 800:

   ```
   H:[someView1]-(value1@900, value2@800)-[someView2]
   ```

4. `someView1` height est égal à height de `someView2` :

```
V:[someView1(==someView2)]
```

Exemple de code



Permet de définir une nouvelle vue, qui comporte un champ de texte et deux boutons ayant des hauteurs égales avec des marges entre eux, et une étiquette en dessous. Les boutons doivent avoir la même largeur et l'étiquette doit déborder à la ligne suivante si le contenu est suffisamment long. Cette vue doit être dimensionnée automatiquement en fonction de son contenu sur les axes horizontaux et verticaux et centrée au milieu de sa vue supérieure.

```
- (void)addView {

    // first lets define a container for our views
    UIView *container = [UIView new];
    // do not forget to disable autoresizing masks for autolayout views
    container.translatesAutoresizingMaskIntoConstraints = NO;
    container.backgroundColor = [UIColor grayColor];

    // now to the subviews. this is mostly boilerplate code:
    UITextField *textField = [UITextField new];
    textField.translatesAutoresizingMaskIntoConstraints = NO;
    textField.borderStyle = UITextBorderStyleRoundedRect;

    UIButton *button1 = [UIButton buttonWithType:UIButtonTypeSystem];
    button1.translatesAutoresizingMaskIntoConstraints = NO;
    [button1 setTitle:@"Send" forState:UIControlStateNormal];

    UIButton *button2 = [UIButton buttonWithType:UIButtonTypeSystem];
    button2.translatesAutoresizingMaskIntoConstraints = NO;
    [button2 setTitle:@"Cancel" forState:UIControlStateNormal];

    UILabel *label = [UILabel new];
    label.translatesAutoresizingMaskIntoConstraints = NO;
    // this line tells the label to let the text overflow to the next line if needed
    label.numberOfLines = 0;
    label.text = @"This label has relatively long text, that should take two lines.";

    // before adding any constraints the views should be present in the hierarchy
    [container addSubview:textField];
    [container addSubview:button1];
    [container addSubview:button2];
    [container addSubview:label];

    // now lets define two helper dictionaries, one for metrics of our view:
    NSDictionary *metrics = @{@"margin": @10, @"textFieldWidth": @160, @"buttonWidth": @44};
    // and the other for view bindings using a handy macro, which effectively creates a
```

```
dictionary with variables of the same name:
    NSDictionary *bindings = NSDictionaryOfVariableBindings(textField, button1, button2,
label);
    // lets define a horizontal format for the first row of views in a variable:
    NSString *horizontalFormat = @"H:|-(margin)-[textField(textFieldWidth)]-(margin)-
[button1(==button2)]-(margin)-[button2]-(margin)-|";
    // this format defines margins of equal size between all views, fixed width for the
textField and sets both buttons to have equal widths
    // lets add these constraints to our container:
    [container addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:horizontalFormat
options:0 metrics:metrics views:bindings]];
    // now lets define horizontal constraints for the second row, where we have the label:
    [container addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:@"H:|-(margin)-
[label]-(margin)-|" options:0 metrics:metrics views:bindings]];
    // another relatively long visual format string:
    NSString *verticalFormat = @"V:|-(margin)-[textField]-(margin)-[label]-(margin)-|";
    // this format string defines vertical constraints for textField and label, and should
also define the height of the container
    // adding these constraints to the container view:
    [container addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:verticalFormat
options:0 metrics:metrics views:bindings]];
    // what we have left are constraints for vertical positions of the buttons
    // lets attach them to the top of the container with a margin:
    [container addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:@"V:|-(margin)-
[button1]" options:0 metrics:metrics views:bindings]];
    [container addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:@"V:|-(margin)-
[button2]" options:0 metrics:metrics views:bindings]];

    // the container is all set up, adding it to the superview:
    [self.view addSubview:container];

    // now lets position our container in its superview
    // you can not use dot notation in the bindings macro, so lets define a temp variable for
the superview:
    UIView *superview = self.view;

    // positioning a view in the center of its superview is not so straightforward
    // we will use a trick from this answer: http://stackoverflow.com/a/14917695/934710
    NSDictionary *containerBindings = NSDictionaryOfVariableBindings(superview, container);
    // width constraint from horizontal format is not part of the trick, but is necessary to
constrain container width
    [self.view addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:@"H:[superview]-
(<=1)-[container(<=superview)]" options:NSLayoutFormatAlignAllCenterY metrics:nil
views:containerBindings]];
    [self.view addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:@"V:[superview]-
(<=1)-[container]" options:NSLayoutFormatAlignAllCenterX metrics:nil
views:containerBindings]];

}
```

Lire Démarrer avec autolayout en ligne: https://riptutorial.com/fr/autolayout/topic/6858/demarrer-avec-autolayout

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec autolayout	Community, Mahesh Agrawal, pckill