EBook Gratis

# APRENDIZAJE autolayout

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#autolayout

# Tabla de contenido

# Acerca de

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

# Capítulo 1: Empezando con autolayout

## Observaciones

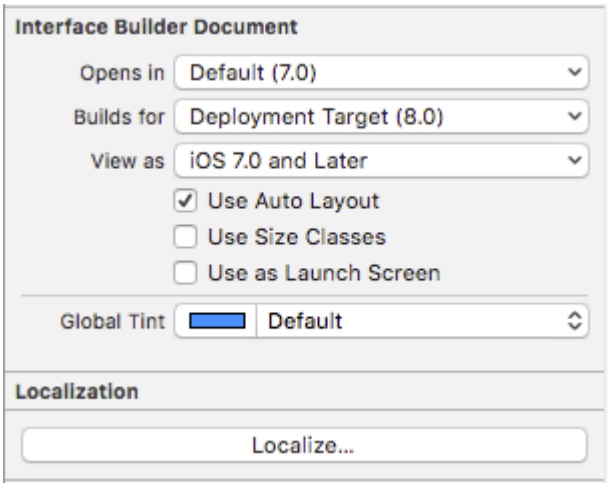
El diseño automático calcula dinámicamente el tamaño y la posición de todas las vistas en su jerarquía de vistas, según las restricciones puestas en esas vistas - por Apple. El diseño automático también puede entenderse describiéndolo como un conjunto de reglas que utiliza para cada una de las vistas en su jerarquía de vistas, que define cómo se realizará la alineación y la reducción o extensión de la vista de acuerdo con los diferentes tamaños de pantalla de los dispositivos.

Deberíamos usar Auto Layout en todos nuestros proyectos porque esto nos ayuda a administrar el diseño de las pantallas para todos los dispositivos disponibles a los que nos dirigimos. Para esto solo necesitamos entender qué es y cómo se realiza y, después de entender cómo funciona, será divertido diseñar un diseño dinámico para múltiples dispositivos.

## Examples

### Instalación o configuración

Para usar el diseño automático necesitamos habilitar una bandera booleana en el guión gráfico. Se muestra en la imagen de abajo. Después de esto podemos usar el diseño automático en nuestro guión gráfico. Hay otra opción si queremos usar clases de tamaño o no. La clase de tamaño también es una opción útil en el diseño automático que nos ayuda a diseñar diferentes tamaños de pantalla de dispositivos de manera diferente con elementos iguales o diferentes. La imagen para la configuración va aquí. Es desde el panel del inspector de archivos del editor de propiedades en Xcode.




### Añadiendo un UIImageView al centro de la pantalla

Si desea agregar un UIImageView al centro de la pantalla con un ancho y alto de 100 píxeles, debe establecer la restricción centro x y el centro y restricción a la vista de campo de

UIImageView y el ancho, la restricción de altura a UIIMageView. Aquí está el código. Hay forma de hacer esto en el guión gráfico, pero he escrito en código porque es comprensible para esta situación en la documentación.

```
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]];
```

### Usando el lenguaje de formato visual

Hay una manera de simplificar la definición de la reproducción automática para vistas usando VFL. Puede parecer difícil al principio, pero en realidad es muy fácil de usar. Algunas definiciones primero:

- `|` representa supervisar
- `H:` o `V:` representa la orientación actual - horizontal o vertical
- Los nombres de las vistas deben estar entre corchetes
- la altura y el ancho de la vista deben estar entre paréntesis
- Los márgenes se especifican entre vistas y rodeados de guiones.
- La prioridad para un margen o tamaño de vista se puede especificar con @

## Ejemplos de sintaxis de formato visual.

1. `someView` se adjunta a los bordes izquierdo y derecho de su superview sin márgenes:

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

2. `someView` se adjunta a la parte superior con un margen de 10 puntos y tiene una altura, igual al `value` :

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

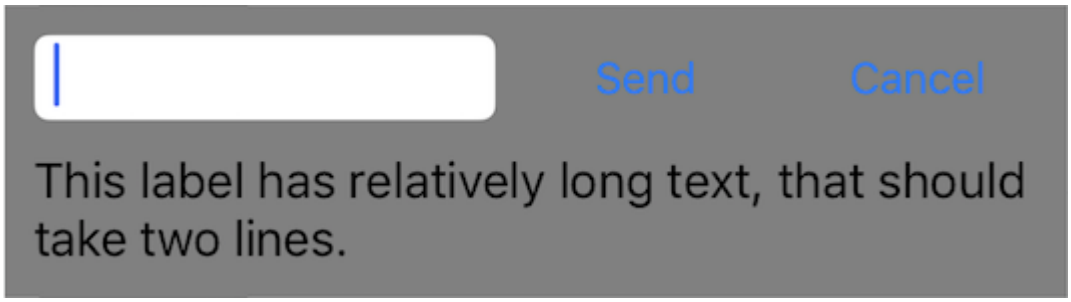
3. `someView1` y `someView2` tienen dos márgenes definidos entre ellos: `value1` con una prioridad de 900, y `value2` con una prioridad de 800:

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

4. `someView1` height es igual a la altura de `someView2` :

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

# Ejemplo de código




Permite definir una nueva vista, que tiene un campo de texto y dos botones con alturas iguales con márgenes entre ellos, y una etiqueta a continuación. Los botones deben tener anchos iguales y la etiqueta debe desbordarse a la siguiente línea si el contenido es lo suficientemente largo. Esta vista debe dimensionarse automáticamente de acuerdo con su contenido tanto en el eje horizontal como en el vertical, y centrarse en el centro de su supervisión.

```
- (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]];

}
```

Lea Empezando con autolayout en línea:
https://riptutorial.com/es/autolayout/topic/6858/empezando-con-autolayout

# Creditos

| S. No | Capítulos | Contributors |
|---|---|---|
| 1 | Empezando con autolayout | Community, Mahesh Agrawal, pckill |