Kostenloses eBook

LERNEN autolayout

Free unaffiliated eBook created from
Stack Overflow contributors.

#autolayout

Inhaltsverzeichnis

Über

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Kapitel 1: Erste Schritte mit Autolayout

Bemerkungen

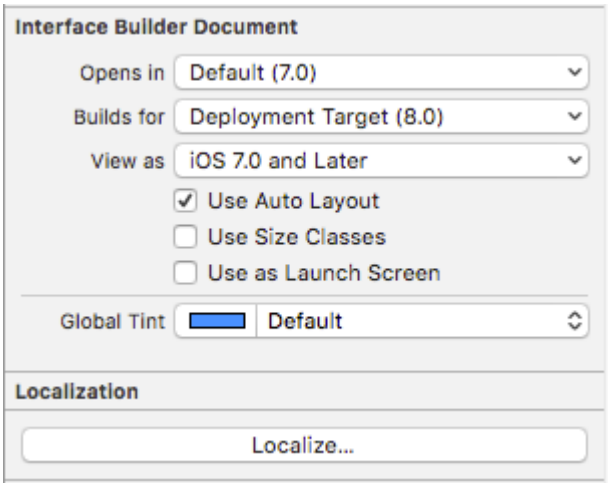
Auto Layout berechnet dynamisch die Größe und Position aller Ansichten in Ihrer Ansichtshierarchie basierend auf den Einschränkungen, die diesen Ansichten auferlegt wurden - von Apple. Das automatische Layout kann auch als Beschreibung eines Satzes von Regeln verstanden werden, die Sie für jede Ansicht in Ihrer Ansichtshierarchie verwenden. Diese Regel definiert, wie das Ausrichten und Verkleinern oder Erweitern der Ansicht entsprechend den verschiedenen Bildschirmgrößen der Geräte durchgeführt wird.

Wir sollten in allen unseren Projekten Auto-Layout verwenden, da dies die Gestaltung der Bildschirme für alle verfügbaren Geräte erleichtert, auf die wir abzielen. Dazu müssen wir nur verstehen, was es ist und wie es funktioniert. Nachdem wir verstanden haben, wie es funktioniert, wird das Entwerfen des dynamischen Layouts für mehrere Geräte Spaß machen.

Examples

Installation oder Setup

Um das automatische Layout verwenden zu können, müssen Sie im Storyboard ein boolesches Flag aktivieren. Es ist in der Abbildung unten dargestellt. Danach können wir das automatische Layout in unserem Storyboard verwenden. Es gibt eine weitere Option, wenn wir Größenklassen verwenden möchten oder nicht. Die Größenklasse ist auch eine hilfreiche Option für das automatische Layout, die uns hilft, unterschiedliche Bildschirmgrößen für Geräte mit gleichen oder unterschiedlichen Elementen unterschiedlich zu gestalten. Das Bild für das Setup geht hier. Es befindet sich im Dateiinspektor des Eigenschaften-Editors in Xcode.



Hinzufügen einer UIImageView zur Bildschirmmitte

Wenn Sie ein UIImageView mit einer Breite und Höhe von 100 Pixel zur Mitte des Bildschirms hinzufügen möchten, müssen Sie die Center X-Einschränkung und die Center Y-Einschränkung

für das Superview von UIImageView und die Breite, Höhe für die UIIMageView festlegen. Hier ist der Code. Es gibt einen Weg, dies in Storyboard zu tun, aber ich habe in Code geschrieben, weil es für diese Situation in der Dokumentation verständlich ist.

```
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]];
```

Visual Format Language verwenden

Es gibt eine Möglichkeit, das Definieren des automatischen Layouts für Ansichten mit VFL zu vereinfachen. Es mag auf den ersten Blick schwer erscheinen, aber es ist wirklich einfach zu bedienen. Einige Definitionen zuerst:

- `|` repräsentiert den Überblick
- `H:` oder `V:` Aktuelle Ausrichtung - horizontal oder vertikal
- Ansichtsnamen sollten in eckigen Klammern stehen
- Ansichtshöhe und -breite sollten in Klammern stehen
- Ränder werden zwischen Ansichten angegeben und von Bindestrichen umgeben
- Priorität für einen Rand oder eine Ansichtsgröße kann mit @ angegeben werden

Beispiele für visuelle Formatsyntax

1. `someView` ist am linken und rechten Rand des Superview ohne Ränder angebracht:

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

2. `someView` wird oben mit einem Rand von 10 Punkten befestigt und hat eine Höhe, die dem `value` :

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

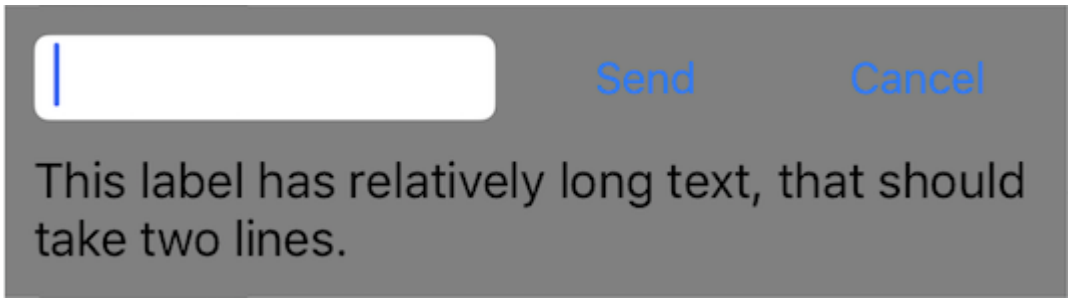
3. `someView1` und `someView2` sind zwei Ränder definiert - `value1` mit einer Priorität von 900 und `value2` mit einer Priorität von 800:

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

4. `someView1` Höhe entspricht der Höhe von `someView2` :

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

Code-Beispiel



Definieren Sie eine neue Ansicht, die ein Textfeld und zwei Schaltflächen mit gleichen Höhen und dazwischen liegenden Rändern sowie eine Beschriftung darunter aufweist. Die Schaltflächen sollten die gleiche Breite haben und die Beschriftung sollte in die nächste Zeile überlaufen, wenn der Inhalt lang genug ist. Diese Ansicht sollte automatisch entsprechend ihrem Inhalt auf der horizontalen und vertikalen Achse skaliert und in der Mitte der Überblendung zentriert werden.

```
- (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]];

}
```

Erste Schritte mit Autolayout online lesen: https://riptutorial.com/de/autolayout/topic/6858/erste-schritte-mit-autolayout

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Autolayout	Community, Mahesh Agrawal, pckill