Auto Layout Without Constraints

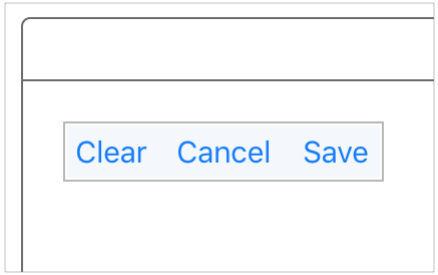
Stack views provide an easy way to leverage the power of Auto Layout without introducing the complexity of constraints. A single stack view defines a row or column of user interface elements. The stack view arranges these elements based on its properties.

- `axis`: (`UIStackView` only) defines the stack view's orientation, either vertical or horizontal.
- `orientation`: (`NSStackView` only) defines the stack view's orientation, either vertical or horizontal.
- `distribution`: defines the layout of the views along the axis.
- `alignment`: defines the layout of the views perpendicular to the stack view's axis.
- `spacing`: defines the space between adjacent views.

To use a stack view, in Interface Builder drag either a vertical or horizontal stack view onto the canvas. Then drag out the content and drop it into the stack.

If an object has an intrinsic content size, it appears in the stack at that size. If it does not have an intrinsic content size, Interface Builder provides a default size. You can resize the object, and Interface Builder adds constraints to maintain its size.

To further fine-tune the layout, you can modify the stack view's properties using the Attributes inspector. For example, the following example uses an 8-point spacing and a Fills Equally distribution.

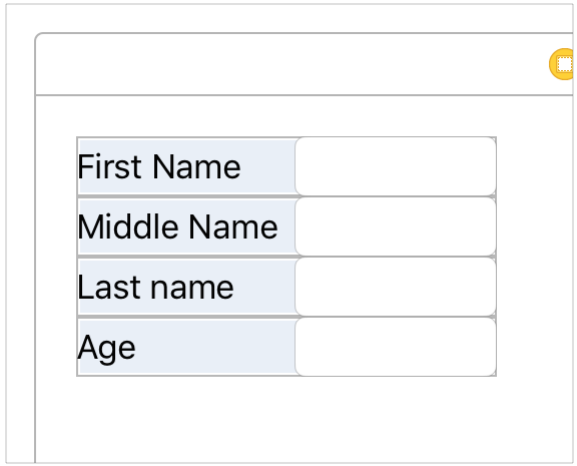


The stack view also bases its layout on the arranged views' content-hugging and compression-resistance priorities. You can modify these using the Size inspector.

NOTE

You can further modify the layout by adding constraints directly to the arranged views; however, you want to avoid any possible conflicts: As a general rule of thumb, if a view's size defaults back to its intrinsic content size for a given dimension, you can safely add a constraint for that dimension. For more information on conflictin

Additionally, you can nest stack views inside other stack views to build more complex layouts.



In general, use stack views to manage as much of your layout as possible. Resort to creating constraints only when you cannot achieve your goals with stack views alone.

For more information on using stack views, see *UIStackView Class Reference* or *NSStackView Class Reference*.

NOTE

Although the creative use of nested stack views can produce complex user interfaces, you cannot completely escape the need for constraints. At a bare minimum, you always need constraints to define the position (and possibly the size) of the outermost stack.