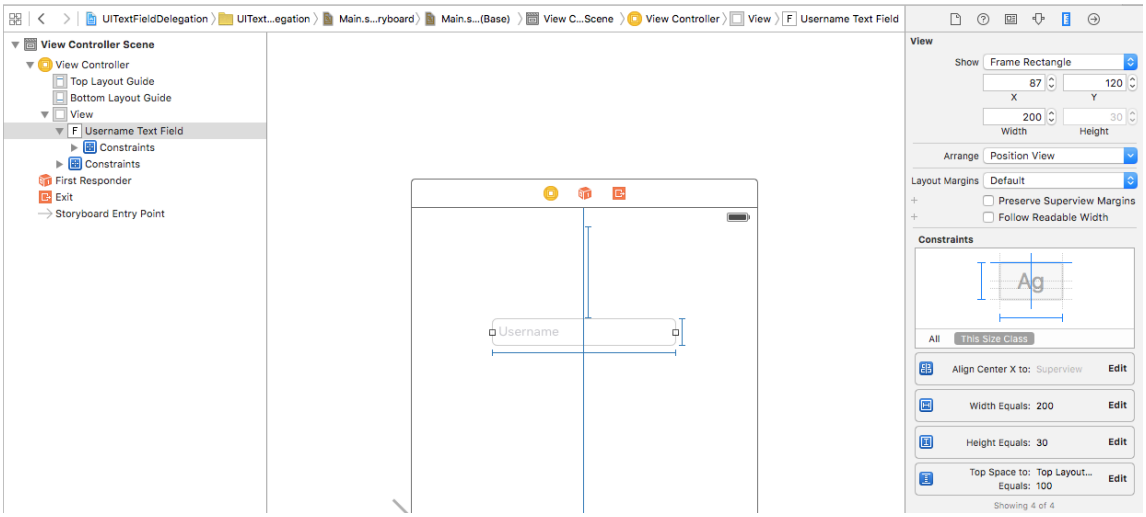
UITextField Delegation with Storyboards

This tutorial will lead you through the creation of a simple app that has a username textfield and a password textfield with the following functionality:

- The password textfield will become active when the Return key is pressed while editing the username textfield
- The keyboard will hide when the Return key is pressed while editing the password textfield
- Text changes in the username textfield will be limited to disallow spaces
- The clear button will allow one-tap clearing of text in the username and password textfields

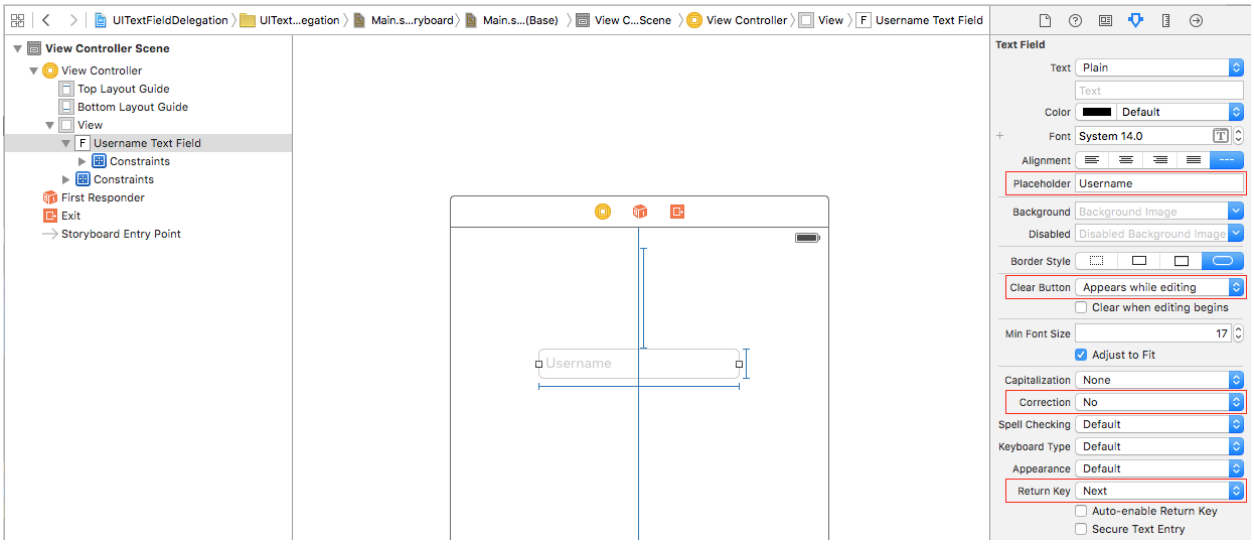
Create a Basic UI

1. Create a new Xcode Project with the Single View template
2. Add the username `UITextField` to your storyboard
    - Configure Auto Layout
        - Horizontally center the textfield in the view controller
        - Offset the textfield 100pts vertically from the Top Layout Guide
        - Set the width to 200pts and the height to 30pts



    - Configure Attributes
        - Add the placeholder text 'Username'
        - Enable the clear button while editing
        - Set the Return key's text to 'Next'

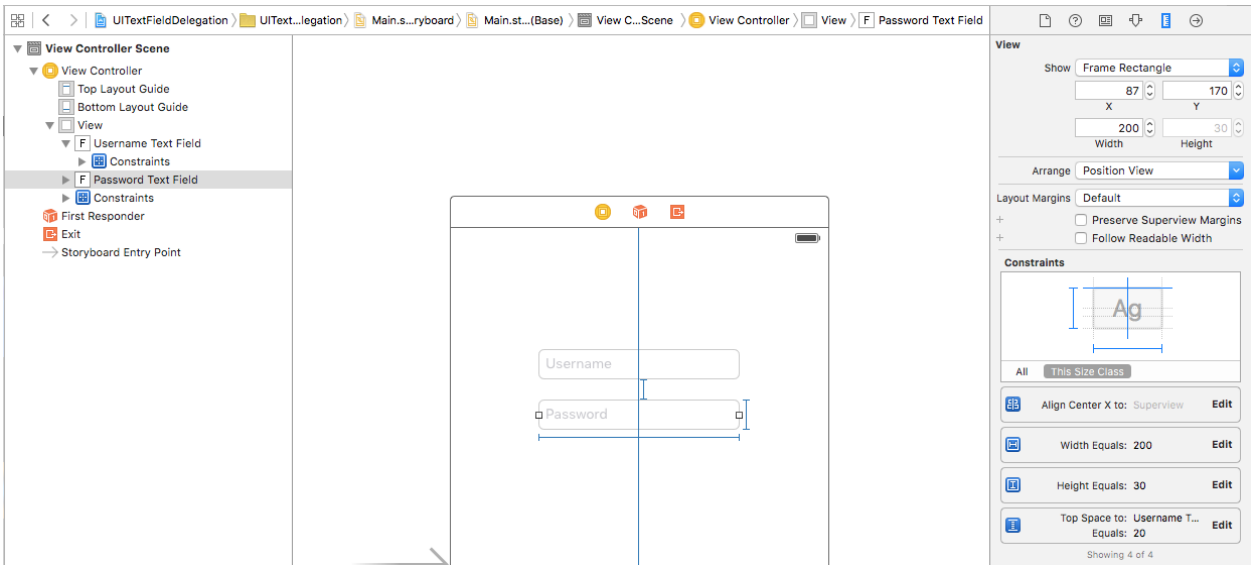
- Disable Auto Correction



3. Add the password `UITextField` to your storyboard

   - Configure Auto Layout

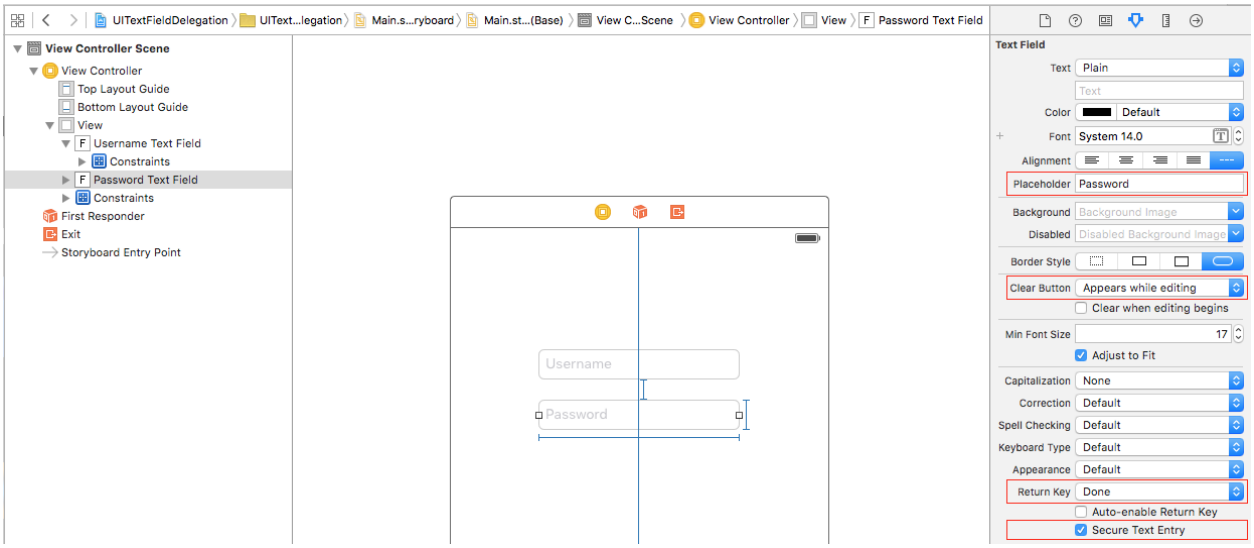
     - Horizontally center the textfield in the view controller
     - Offset the password textfield 20pts below from the username textfield
     - Set the width to 200pts and the height to 30pts



   - Configure Attributes

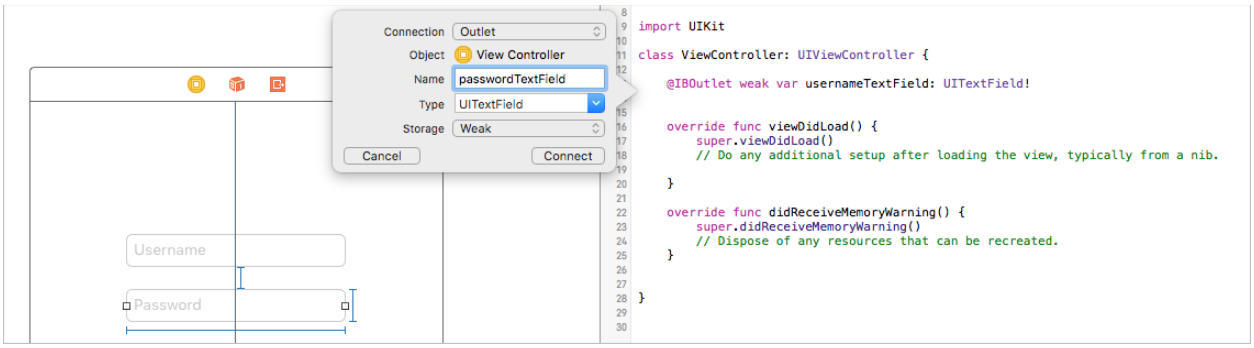
     - Add the placeholder text 'Password'

- Enable the clear button while editing
- Set the Return key's text to 'Done'
- Enable Secure Text Entry to display bullets while entering text



Add Outlets and Delegates

1. Add outlet properties for your textfields
    - Add a weak `UITextField` outlet for your username textfield called `usernameTextField`
    - Add a weak `UITextField` outlet for your password textfield called `passwordTextField`

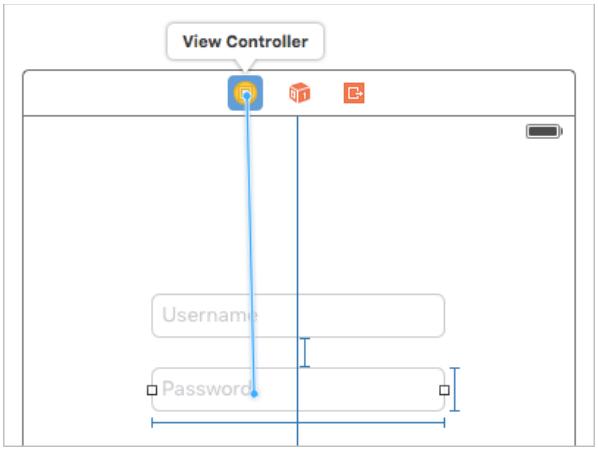


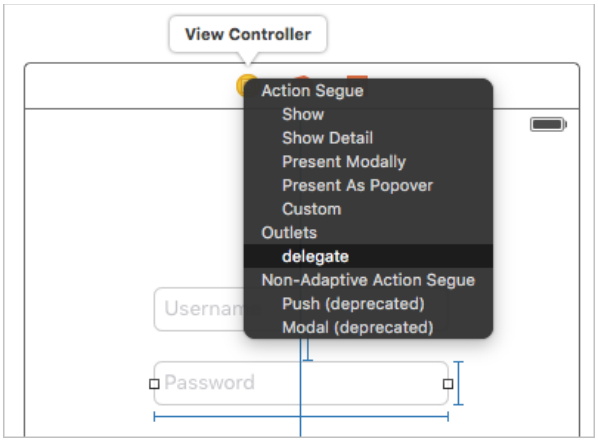
2. Add your ViewController class as the delegate for your textfields
    - Have your ViewController class adopt the `UITextFieldDelegate` protocol
    - Hook up delegates in your storyboard
        - In your storyboard, Control-drag from usernameTextField to your ViewController, and select delegate

```swift

import UIKit

class ViewController: UIViewController, UITextFieldDelegate {

    @IBOutlet weak var usernameTextField: UITextField!
    @IBOutlet weak var passwordTextField: UITextField!


    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.

    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


}


```

- In your storyboard, Control-drag from passwordTextField to your ViewController, and select delegate

○ This will set your ViewController class as the delegate property for both textfields





Add Delegate Methods

1. Add and test delegate methods
    - Add all of the methods defined by the `UITextFieldDelegate` protocol to your ViewController class
    - Add return values to methods that require a return value (the default for all of them should be `true`)
    - Add a print statement to each method so you can verify they are getting called at the appropriate time
    - Run your app in the simulator and make sure your delegate methods are getting called

```
import UIKit

class ViewController: UIViewController, UITextFieldDelegate {

    @IBOutlet weak var usernameTextField: UITextField!
    @IBOutlet weak var passwordTextField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.

    }

    func textFieldShouldBeginEditing(textField: UITextField) -> Bool {
        print("textFieldShouldBeginEditing")
        return true
    }

    func textFieldDidBeginEditing(textField: UITextField) {
        print("textFieldDidBeginEditing")
    }

    func textField(textField: UITextField, shouldChangeCharactersInRange range: NSRange, replacementString string: String) -> Bool {
        print("textFieldShouldChangeCharactersInRange")
        return true
    }

    func textFieldShouldReturn(textField: UITextField) -> Bool {
        print("textFieldShouldReturn")
        return true
    }

    func textFieldShouldClear(textField: UITextField) -> Bool {
        print("textFieldShouldClear")
        return true
    }

    func textFieldShouldEndEditing(textField: UITextField) -> Bool {
        print("textFieldShouldEndEditing")
        return true
    }

    func textFieldDidEndEditing(textField: UITextField) {
        print("textFieldDidEndEditing")
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

}
```

2. Configure delegate methods
    - While editing `usernameTextfield`, we want to make `passwordTextfield` become active when the Return key is pressed
        - You can force a textfield to gain focus by calling the `becomeFirstResponder()` method of a textfield
        - You can compare instances of `UIView` subclasses using the `isEqual(_:)` method
        - In the `textFieldShouldReturn(_:)` delegate method, check if the textField being edited is `usernameTextField`, and if so, then call `becomeFirstResponder()` on `passwordTextField`

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
    print("textFieldShouldReturn")

    if textField.isEqual(self.usernameTextField) {
        passwordTextField.becomeFirstResponder()
    }

    return true
}
```

    - While editing `passwordTextfield`, we want to make the keyboard hide when the Return key is pressed
        - You can force a textfield to lose focus and become inactive by calling the `resignFirstResponder()` method of a textfield - this will also hide the keyboard in the process
        - In the `textFieldShouldReturn(_:)` delegate method, add another check to see if the textField being edited is `passowrdTextField`, and if so, call `resignFirstResponder()` on `passwordTextField`, thereby hiding the keyboard

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
    print("textFieldShouldReturn")

    if textField.isEqual(self.usernameTextField) {
        passwordTextField.becomeFirstResponder()
    }
    else if textField.isEqual(self.passwordTextField) {
        passwordTextField.resignFirstResponder()
    }

    return true
}
```

- We want to prevent spaces from being added to `usernameTextField`
  - In the `textField(_:shouldChangeCharactersInRange:replacementString:)` delegate method, check if the active textfield is `usernameTextField`, and if so, make sure the method only returns `true` if the `replacementString` is not a space

```
func textField(textField: UITextField, shouldChangeCharactersInRange range: NSRange, replacementString string: String) -> Bool {
    print("textFieldShouldChangeCharactersInRange")

    if string == " " {
        return false
    }
    else {
        return true
    }
}
```

- We want to enable one-tap clearing of text in the username and password textfields
  - Simply return `true` in the `textFieldShouldClear(_:)` delegate method to enable the default handling when the clear button is tapped

```
func textFieldShouldClear(textField: UITextField) -> Bool {
    print("textFieldShouldClear")
    return true
}
```

Test the App

1. Run your app in the simulator

2. Test `usernameTextField`
    - Add text to `usernameTextField` and you should see a clear button - tap it and all the text should clear
    - If you try adding a space to `usernameTextField`, nothing should happen
    - Press the Return key on the keyboard when `usernameTextField` is selected, and now `passwordTextField`should be selected
3. Test `passwordTextField`
    - If you press the Return key while editing `passwordTextField`, the keyboard should hide