

# Разработка iOS приложений

Лекция 4.

Графический интерфейс  
пользователя

# Сегодня

---

- Паттерн Model View Controller
- UIView, UIViewController
- InterfaceBuilder

# ModelViewController

---

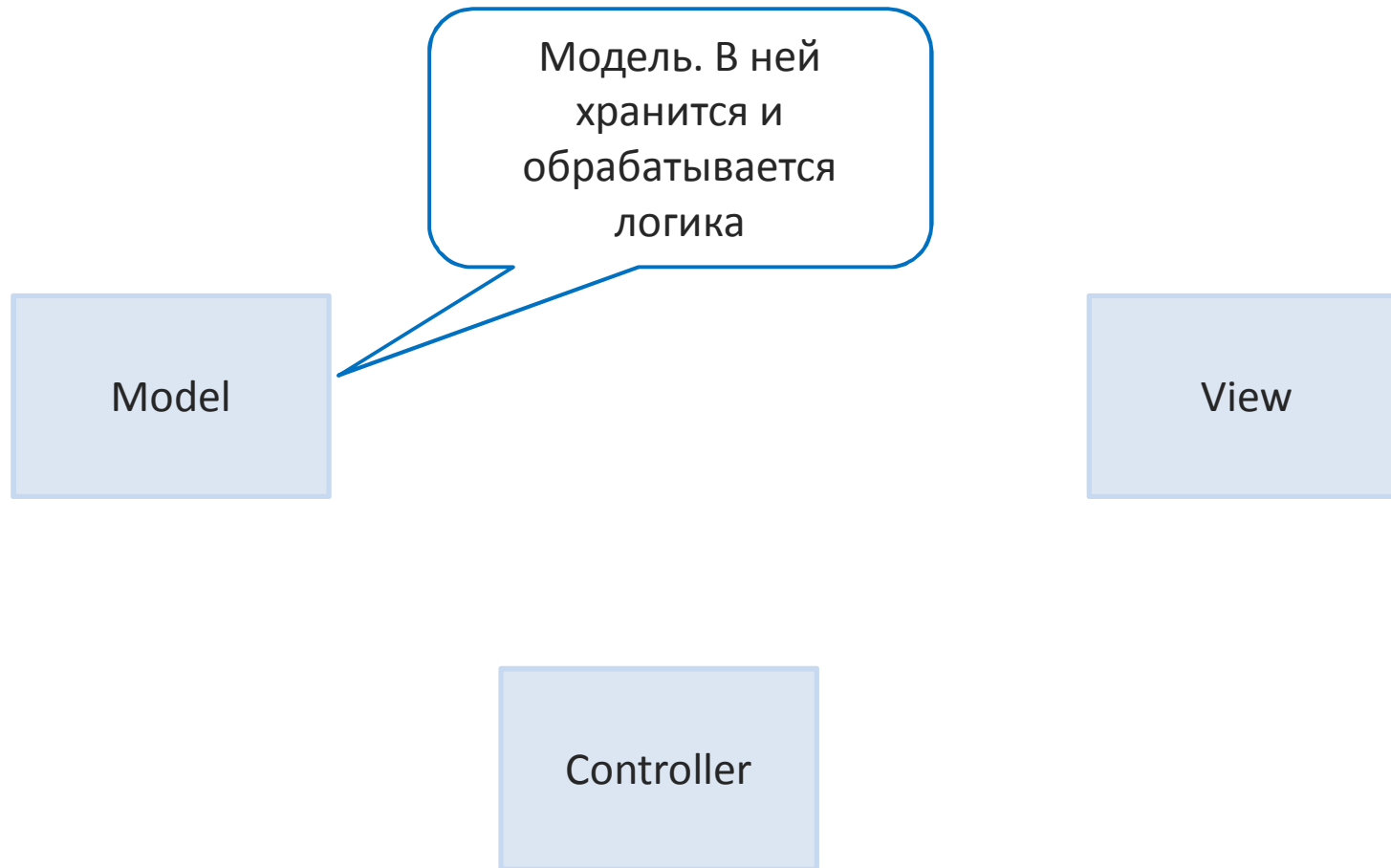
Model

View

Controller

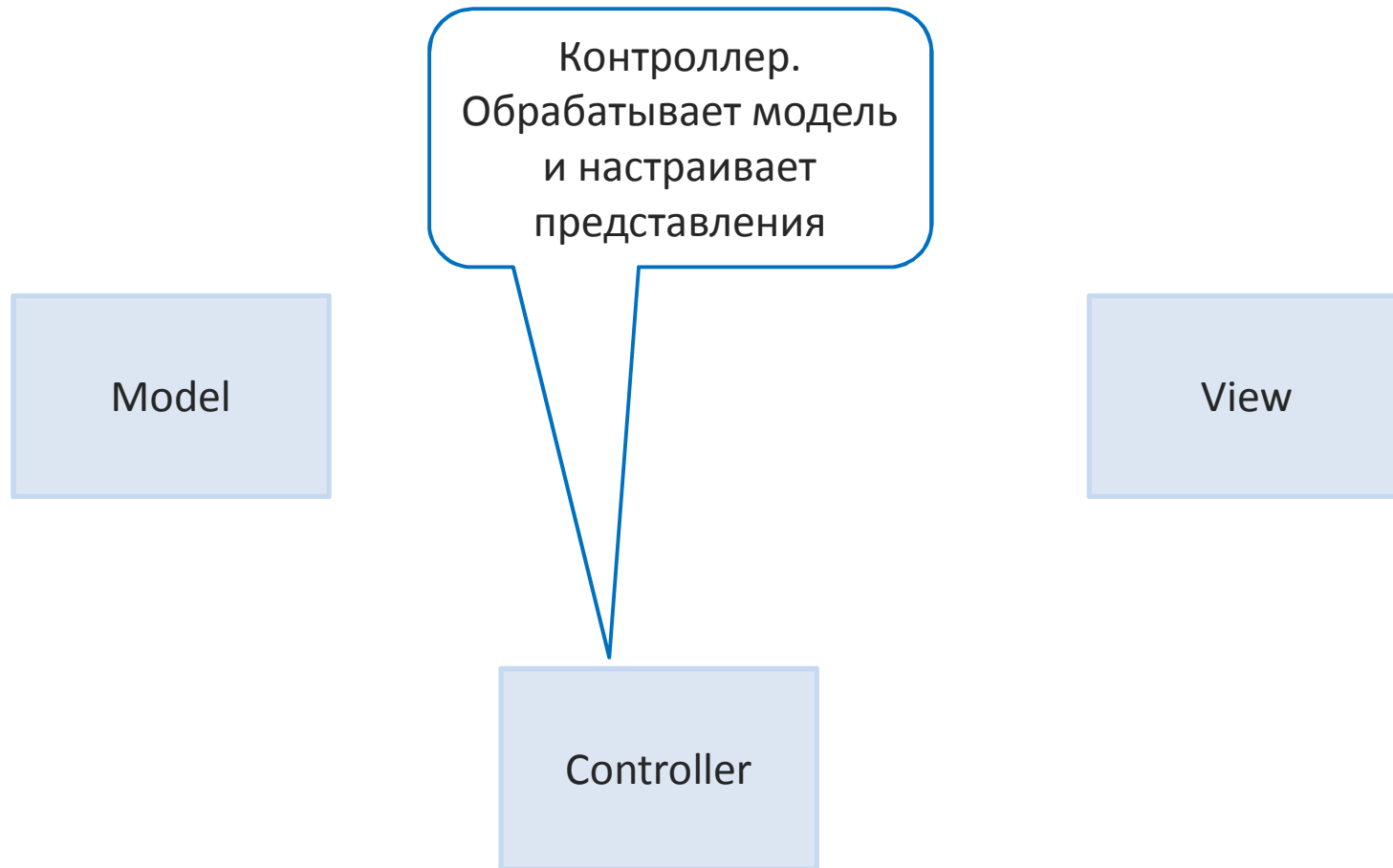
# ModelViewController

---



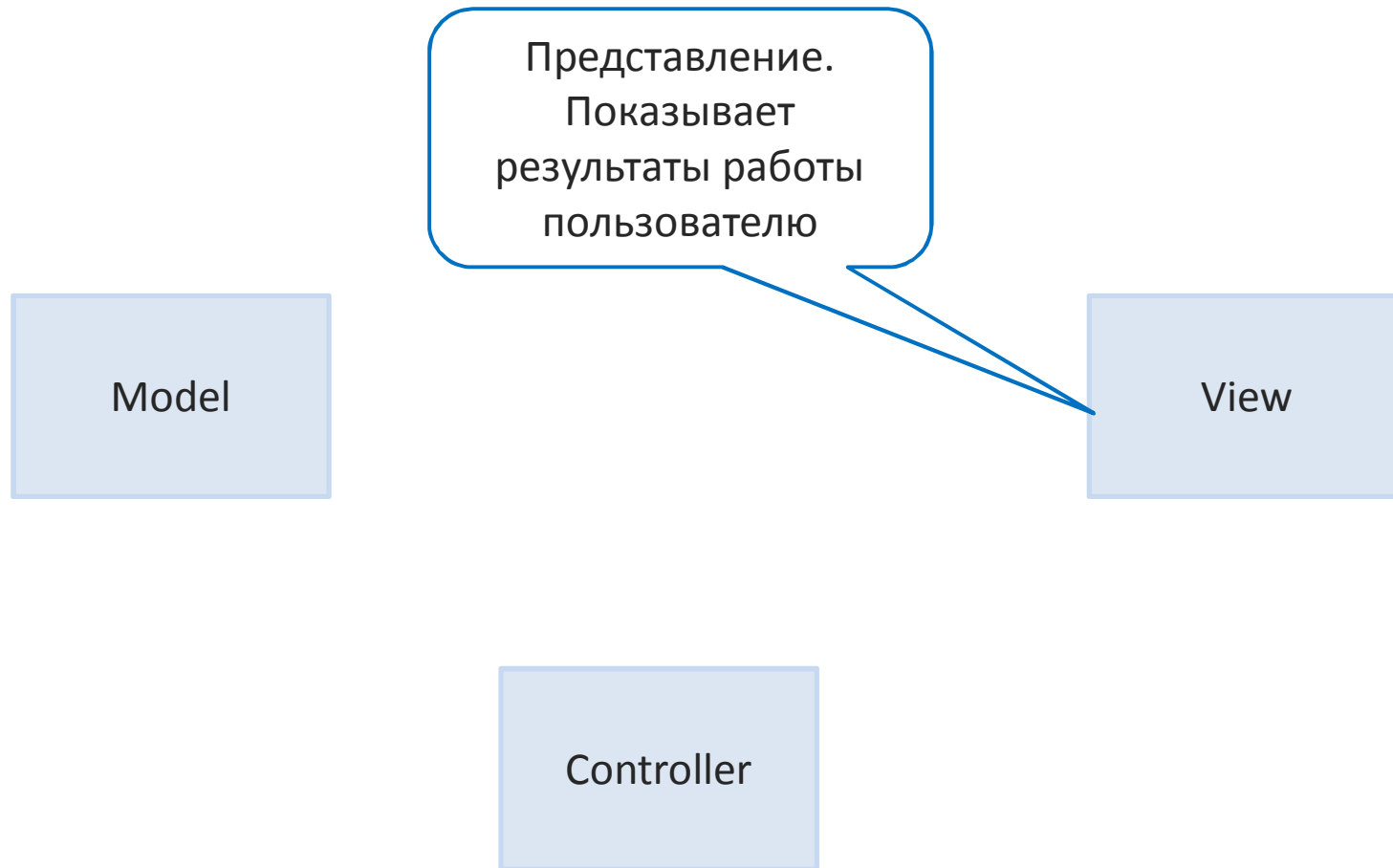
# ModelViewController

---



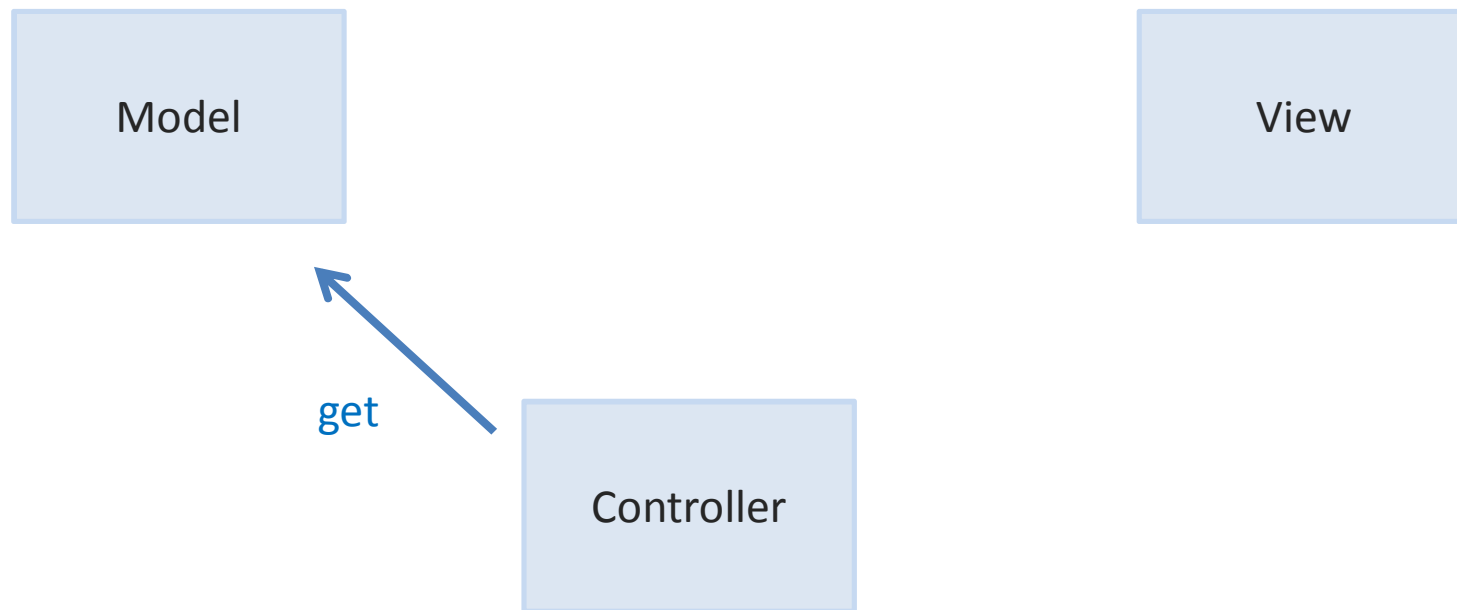
# ModelViewController

---



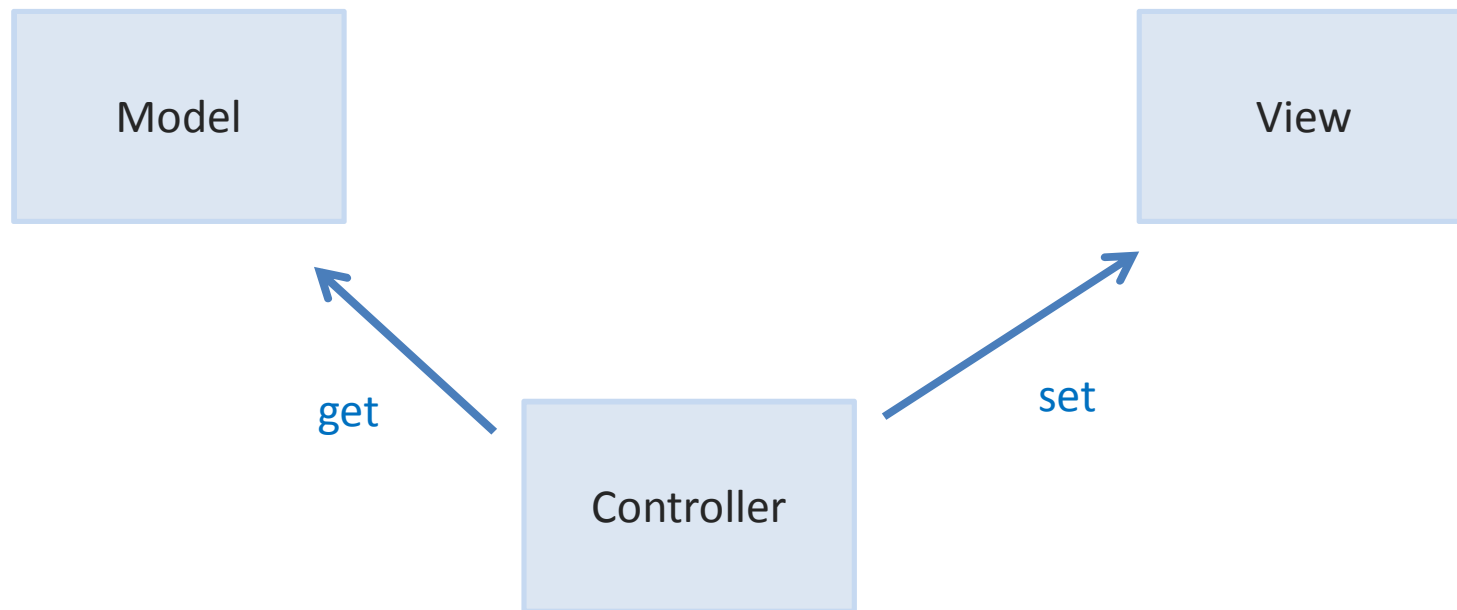
# ModelViewController

---



# ModelViewController

---





# ModelViewController

---

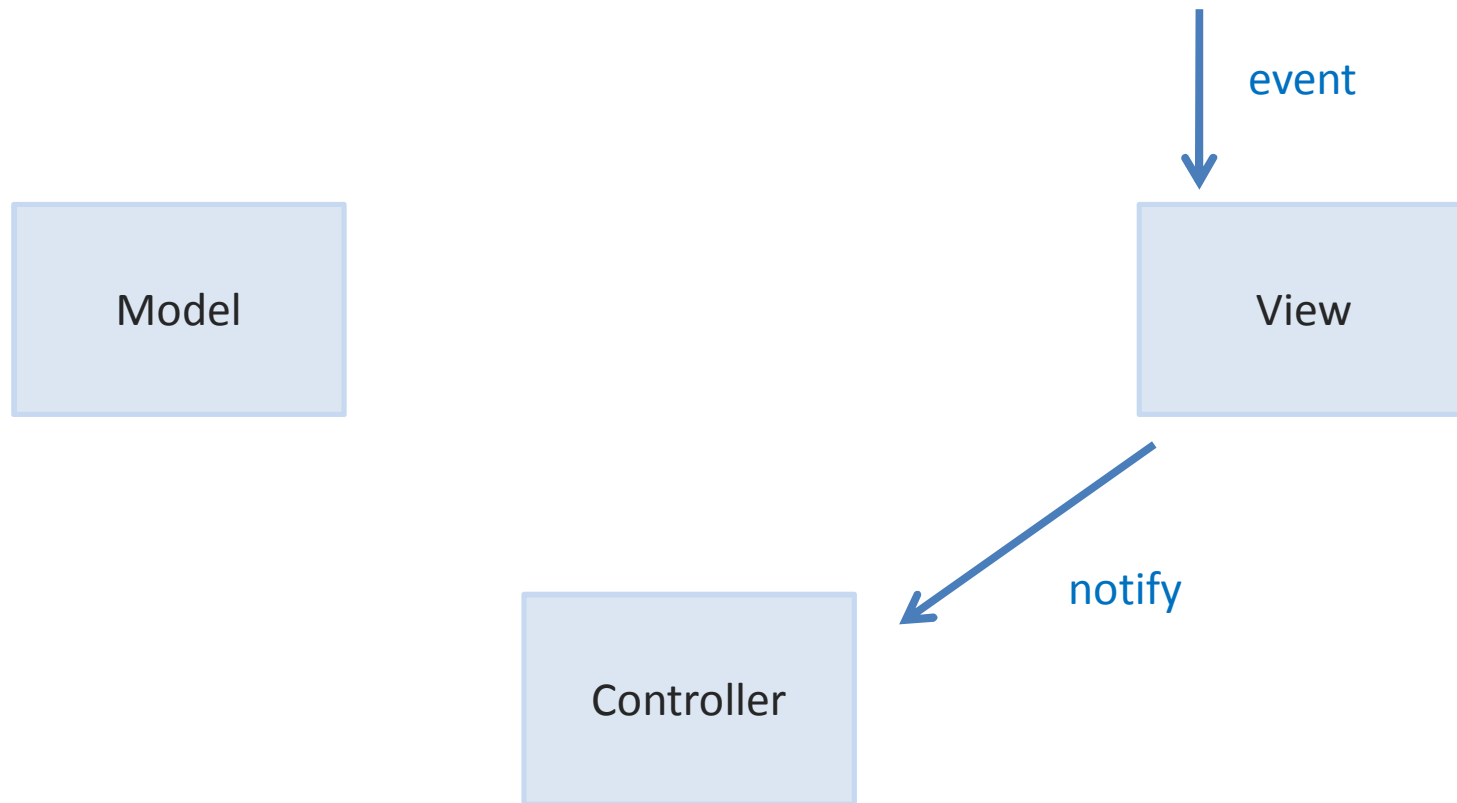
Model

View

Controller

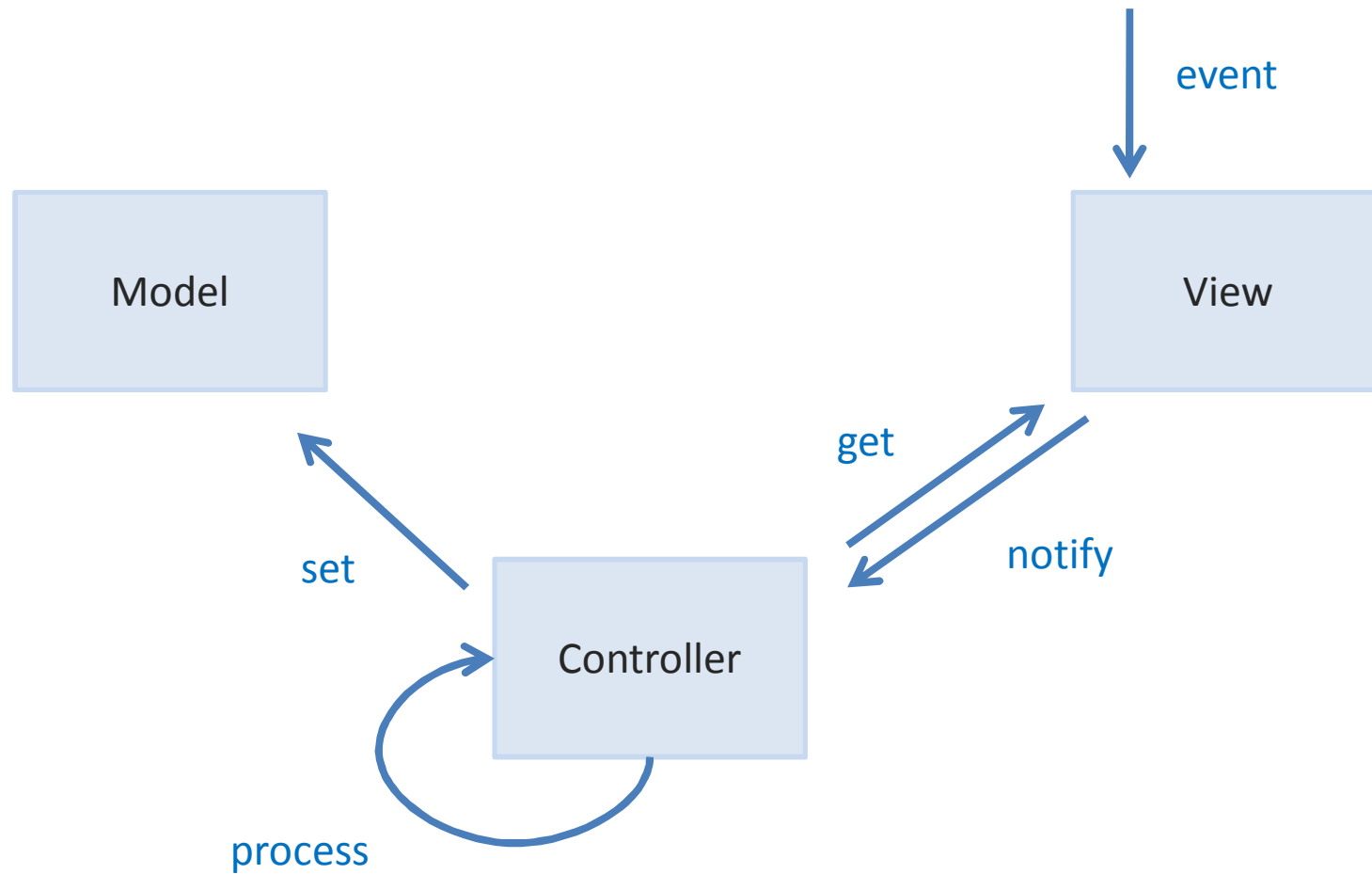
# ModelViewController

---



# ModelViewController

---



# ModelViewController

---

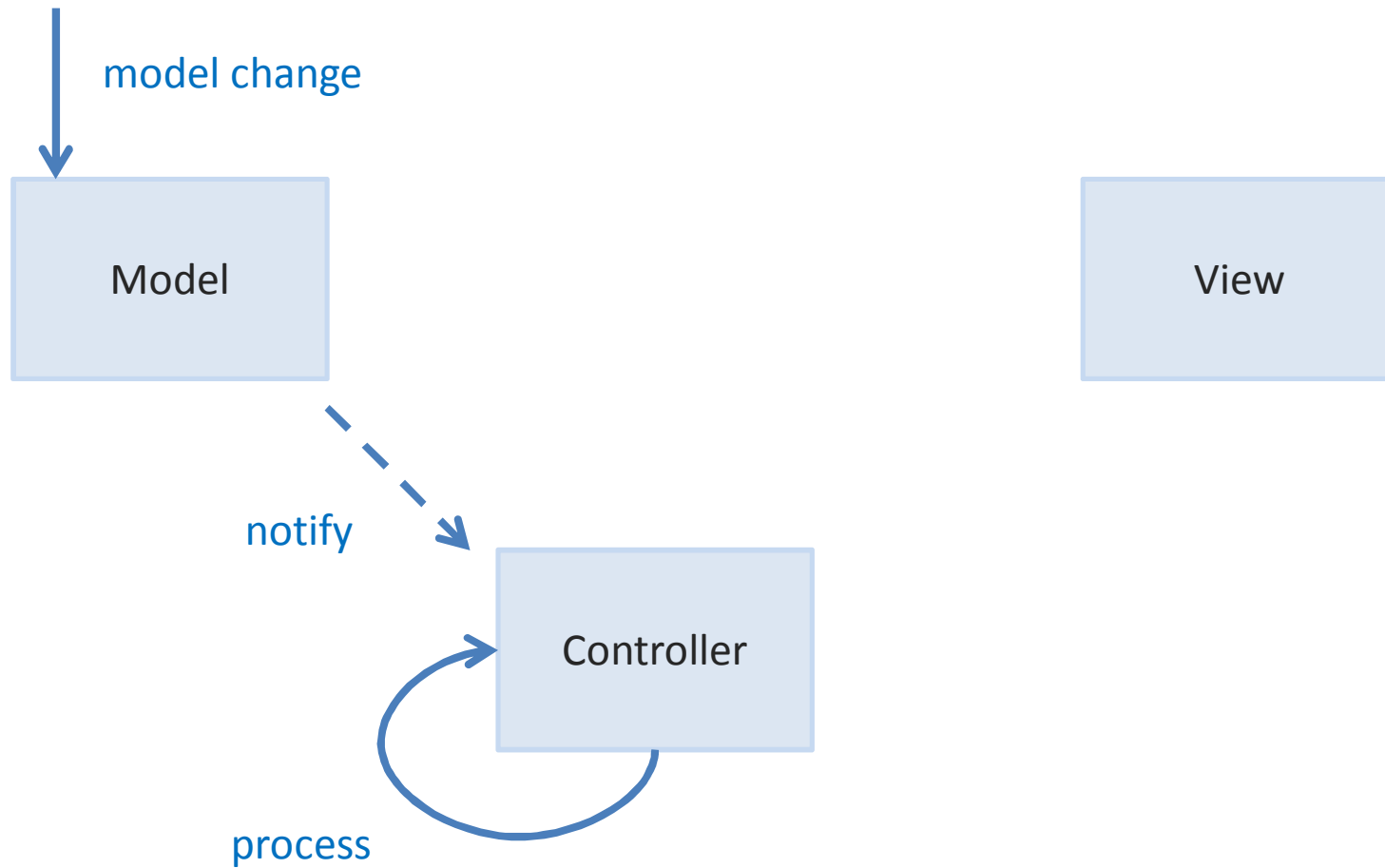
Model

View

Controller

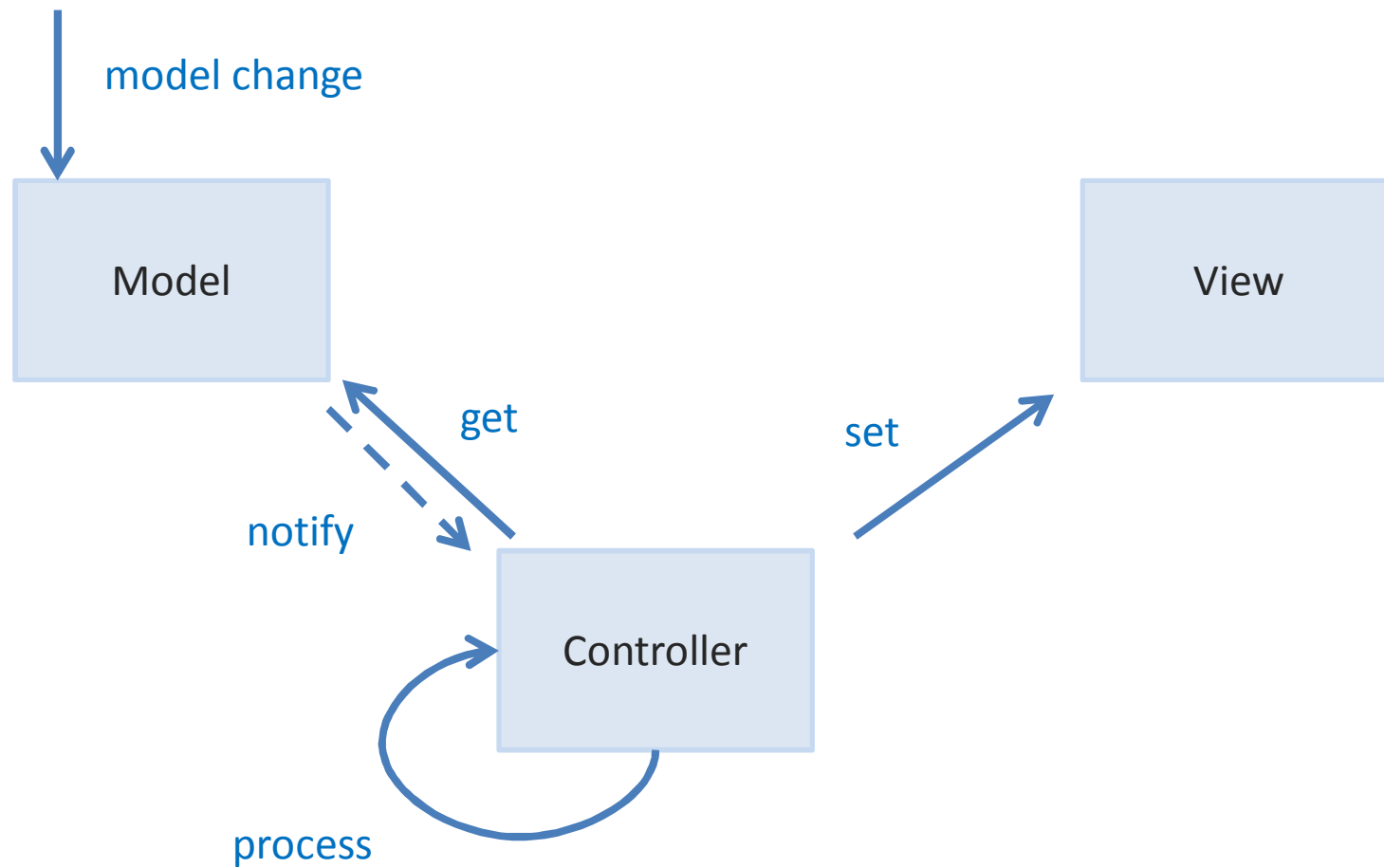
# ModelViewController

---



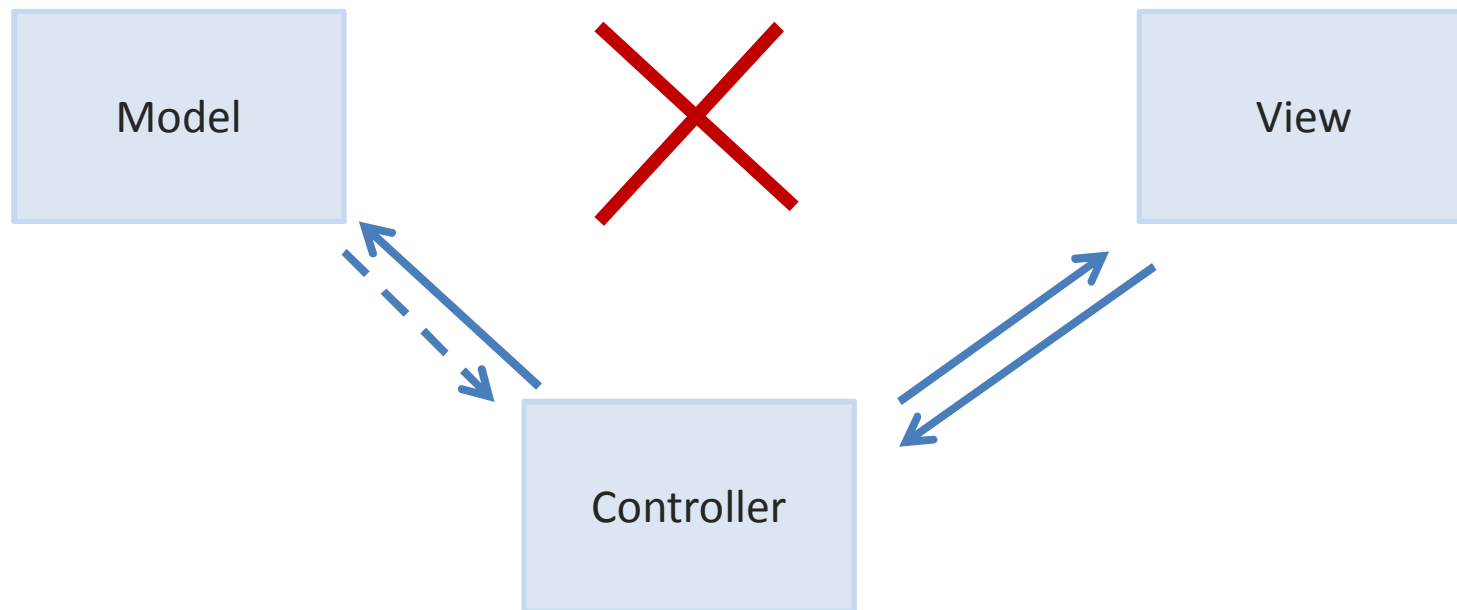
# ModelViewController

---



# ModelViewController

---



# Cocoa Touch

---

- Модель – Rss, RssChannel, RssItem, NSDictionary, ...
- Контроллер - UIViewController
- Представление - UIView



# Представление (UIView)

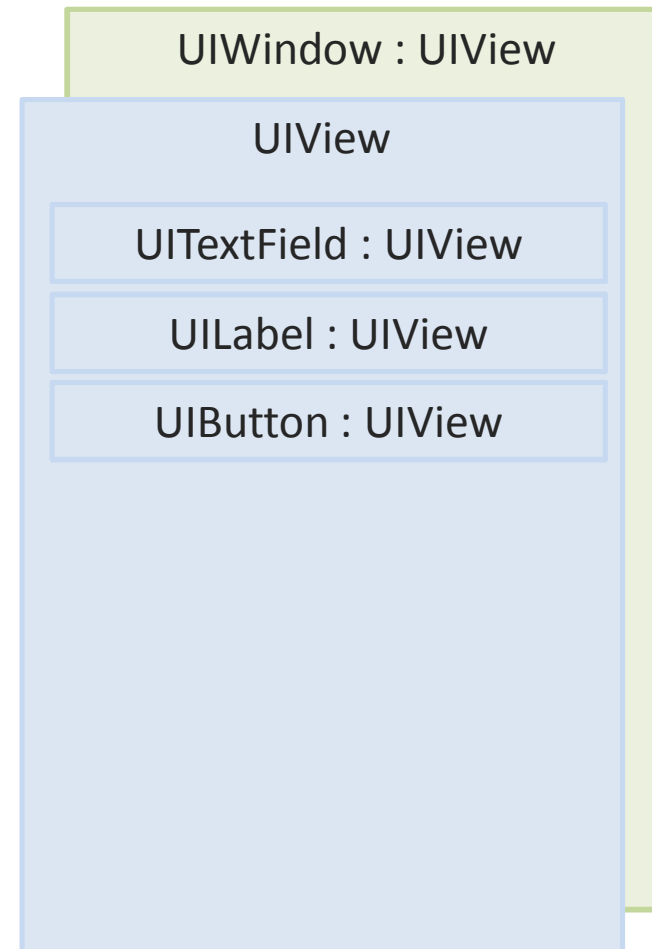
---

UIView

- NSArray\* subviews
- UIView\* superview
- -( void ) addSubview: ( UIView\* )view
- -( void ) removeFromSuperview

UIWindow : UIView

Контролы – см. MobileHIG



# Представление – управление памятью

---

- `superview` всегда делает `retain` на своих `subviews`
- При удалении `view` имейте ввиду то, что при `removeFromSuperview` вызывается `release`. Поэтому если `view` Вам нужна – сделайте себе `retain`.
- `IBOutlet`'ы `retain`'аться. Однако мы должны сделать им `release` в нашем `dealloc`
- `IBOutlet`'ы лучше делать `@property` нежели переменными экземпляра

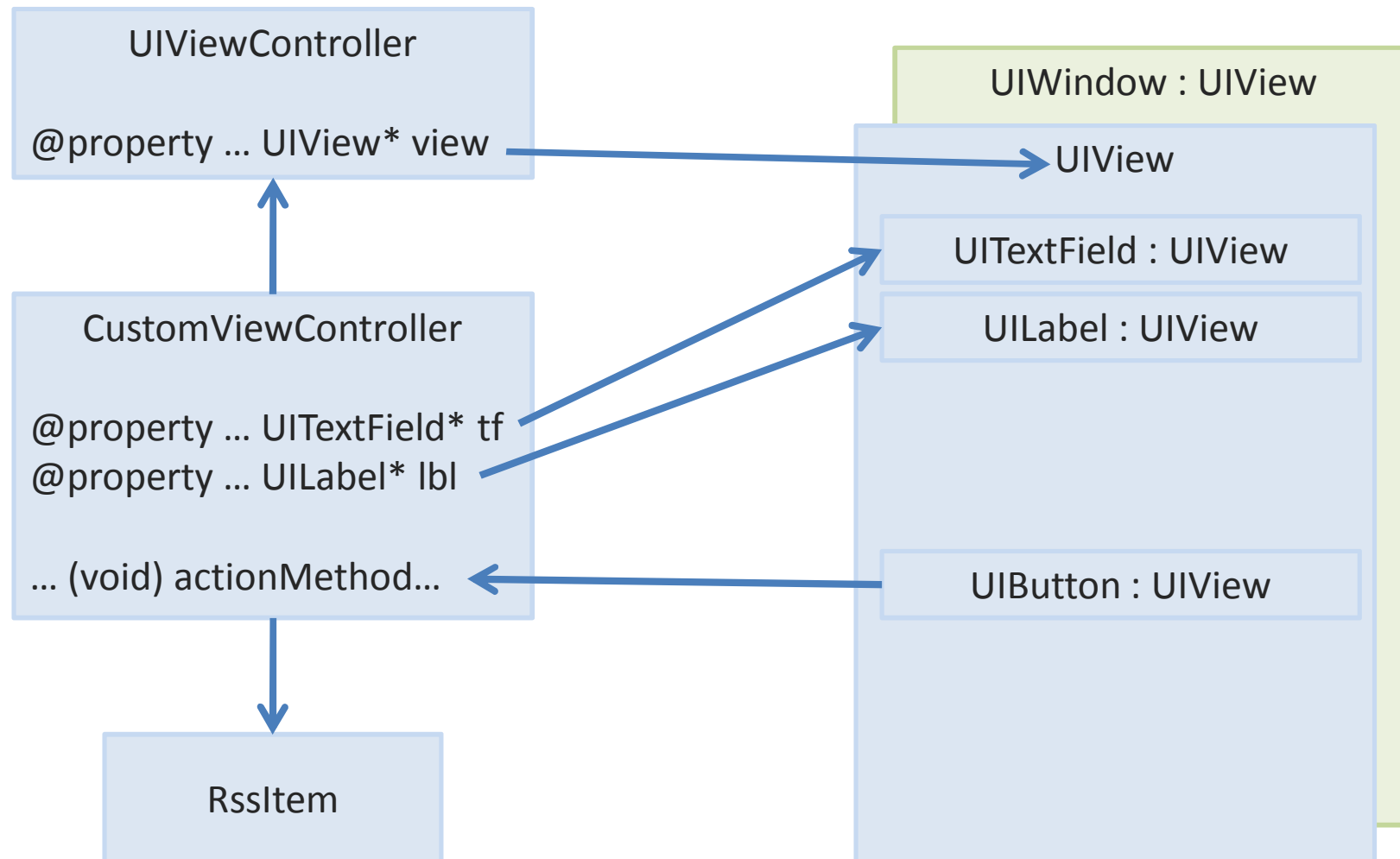
# Что должен делать контроллер

---

- Создать View
- Создать еще View и поместить их в subviews
- Получить доступ к модели, получить из нее необходимые данные
- Проинициализировать элементы управления в View необходимым образом
- Подписаться на события во View и корректно их обрабатывать
- При обработке события соответствующим образом актуализировать содержание контролов

# Контроллер (UIViewController)

---



# Важные методы UIViewController

---

- `viewDidLoad` – представление загружено
- `viewDidUnload` – представление выгружено
- `viewDidAppear:animated:` – представление показано на экране
- `viewDidDisappear:animated:` – представление убрано с экрана
  
- `viewWillAppear:animated:` - представление будет показано на экране
- `viewWillDisappear:animated:` - представление будет убрано с экрана

# Работа через Interface Builder

---

## Code

CustomViewController

IBOutlet ... firstTextView  
IBOutlet ... secondTextView

## Interface Builder

File's Owner

View

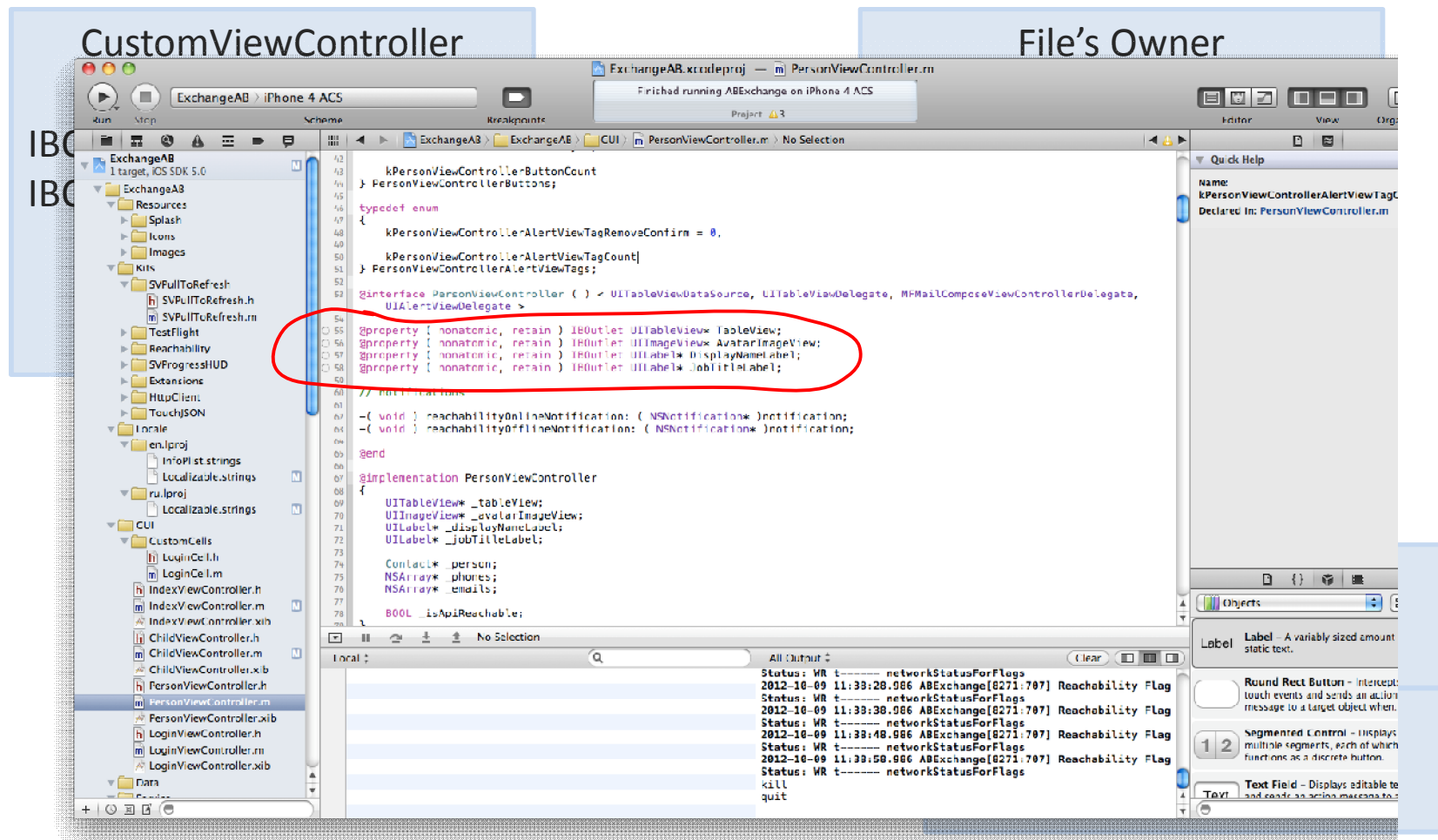
firstTextView

SecondTextView

# Работа через Interface Builder

Code

Interface Builder



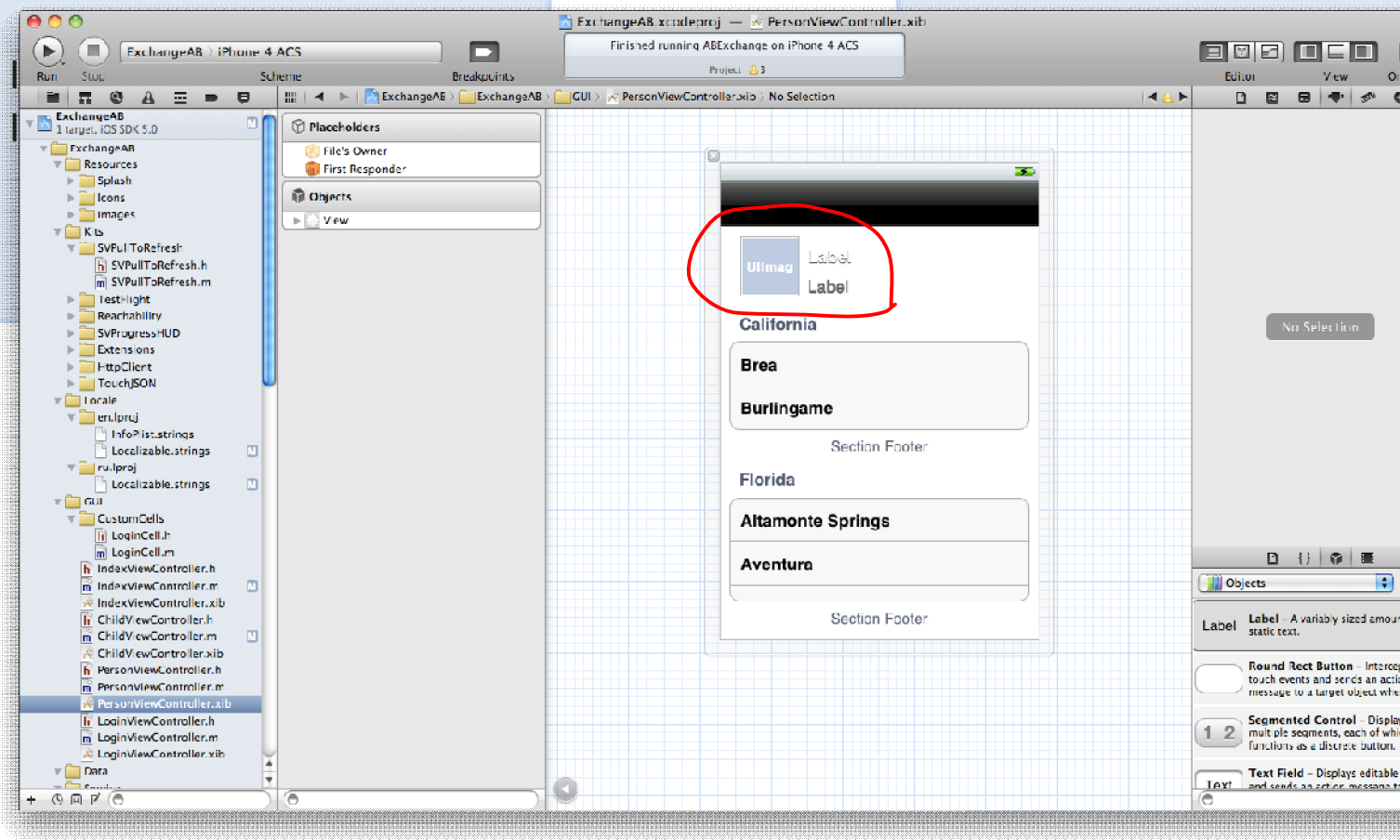
# Работа через Interface Builder

Code

CustomViewController

Interface Builder

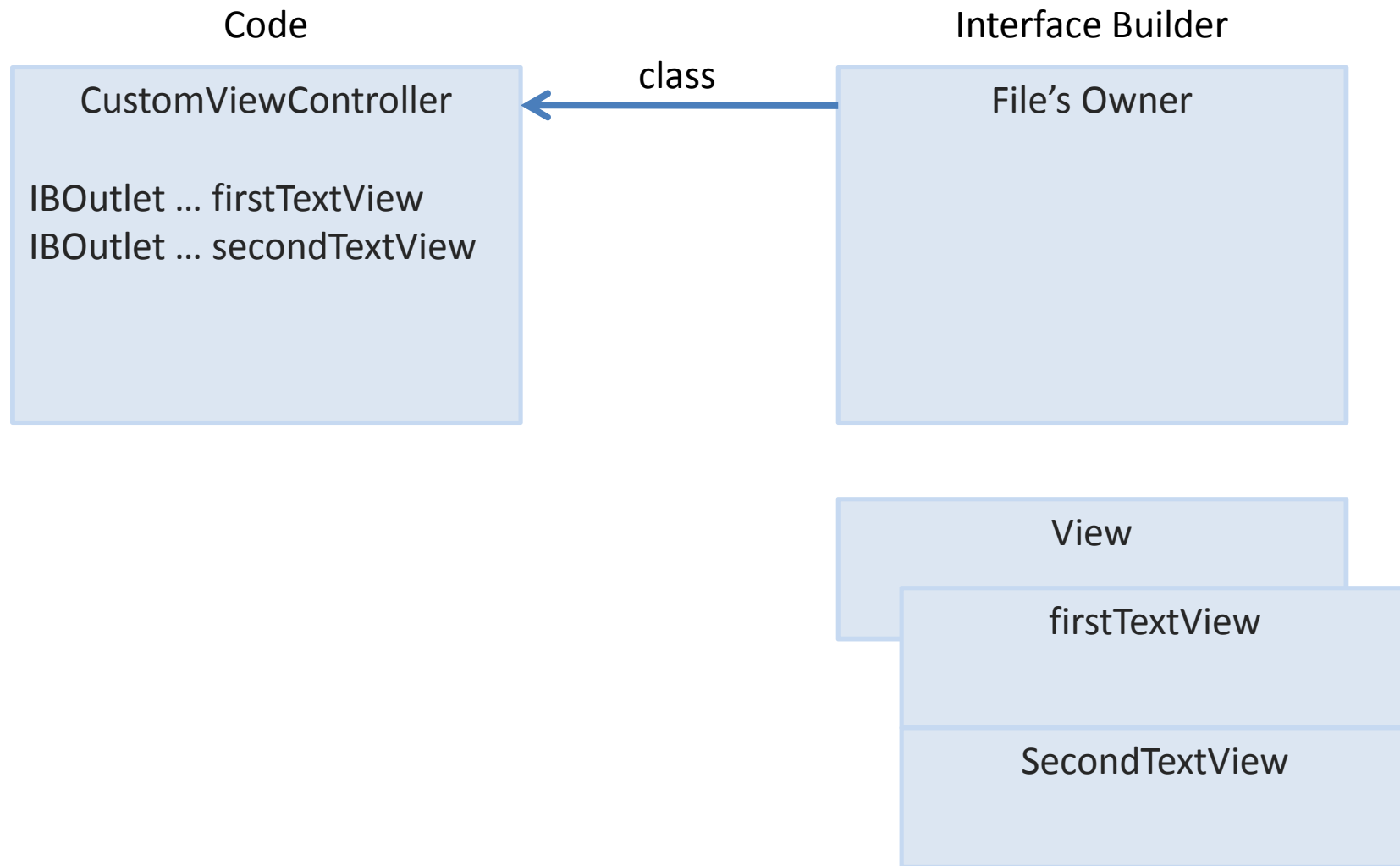
File's Owner





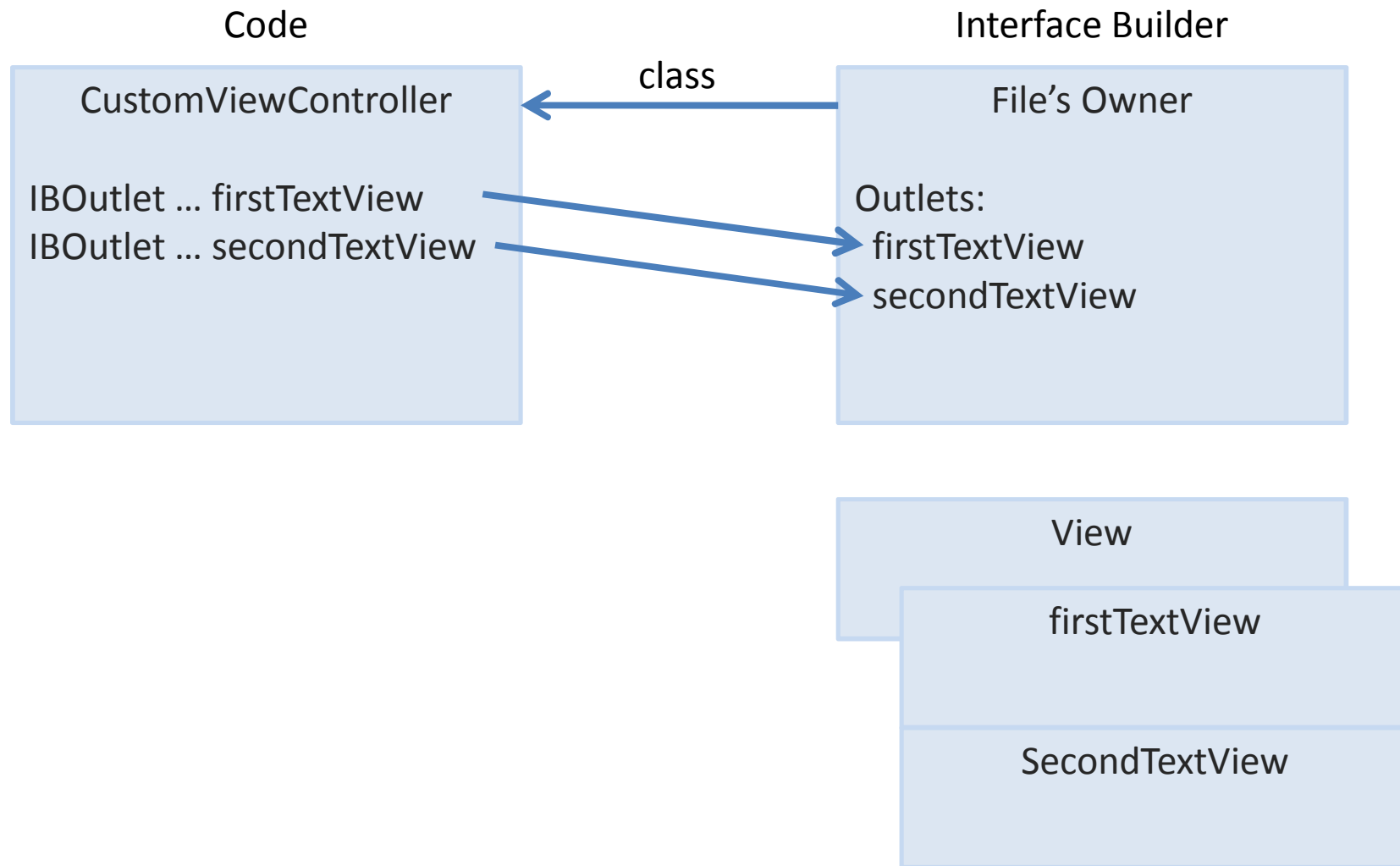
# Работа через Interface Builder

---



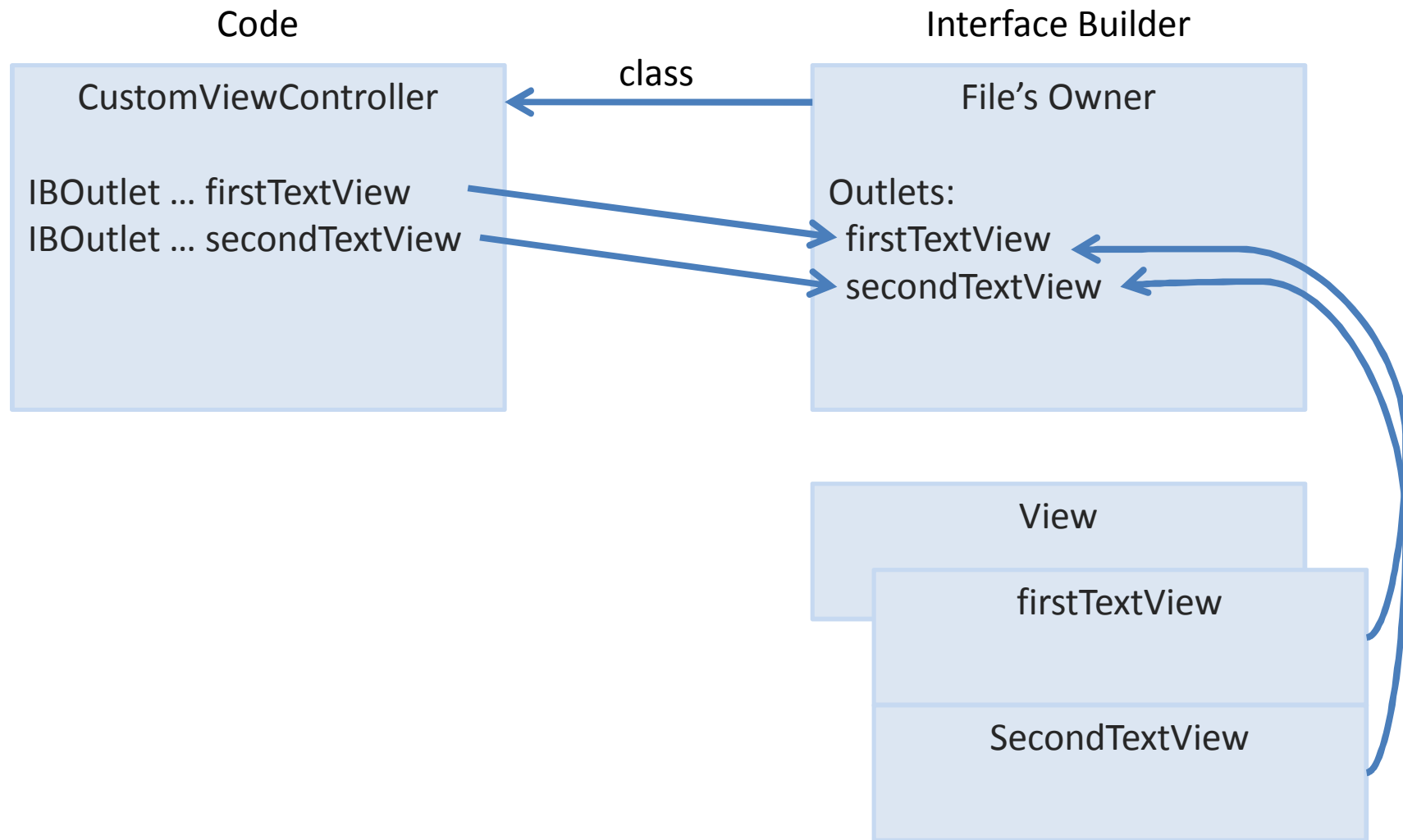
# Работа через Interface Builder

---



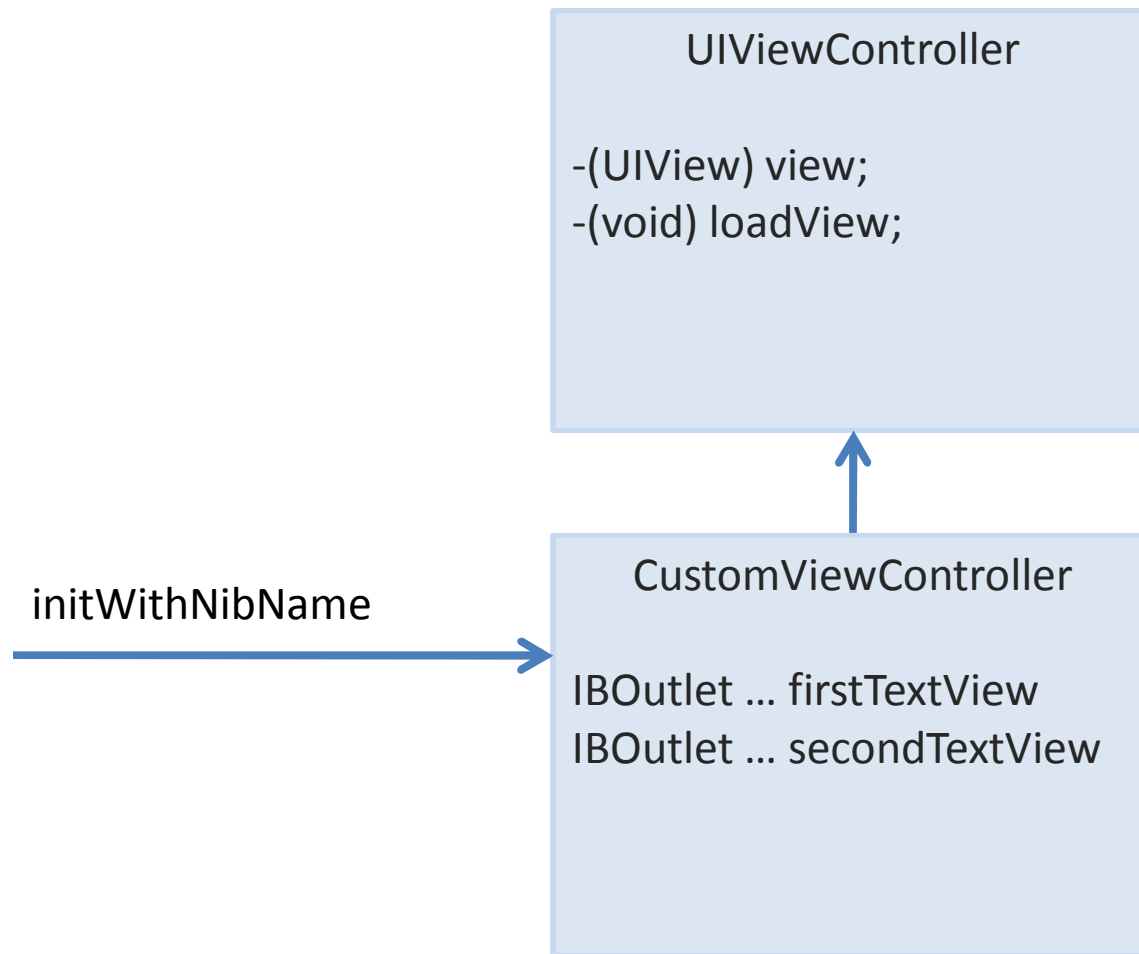
# Работа через Interface Builder

---



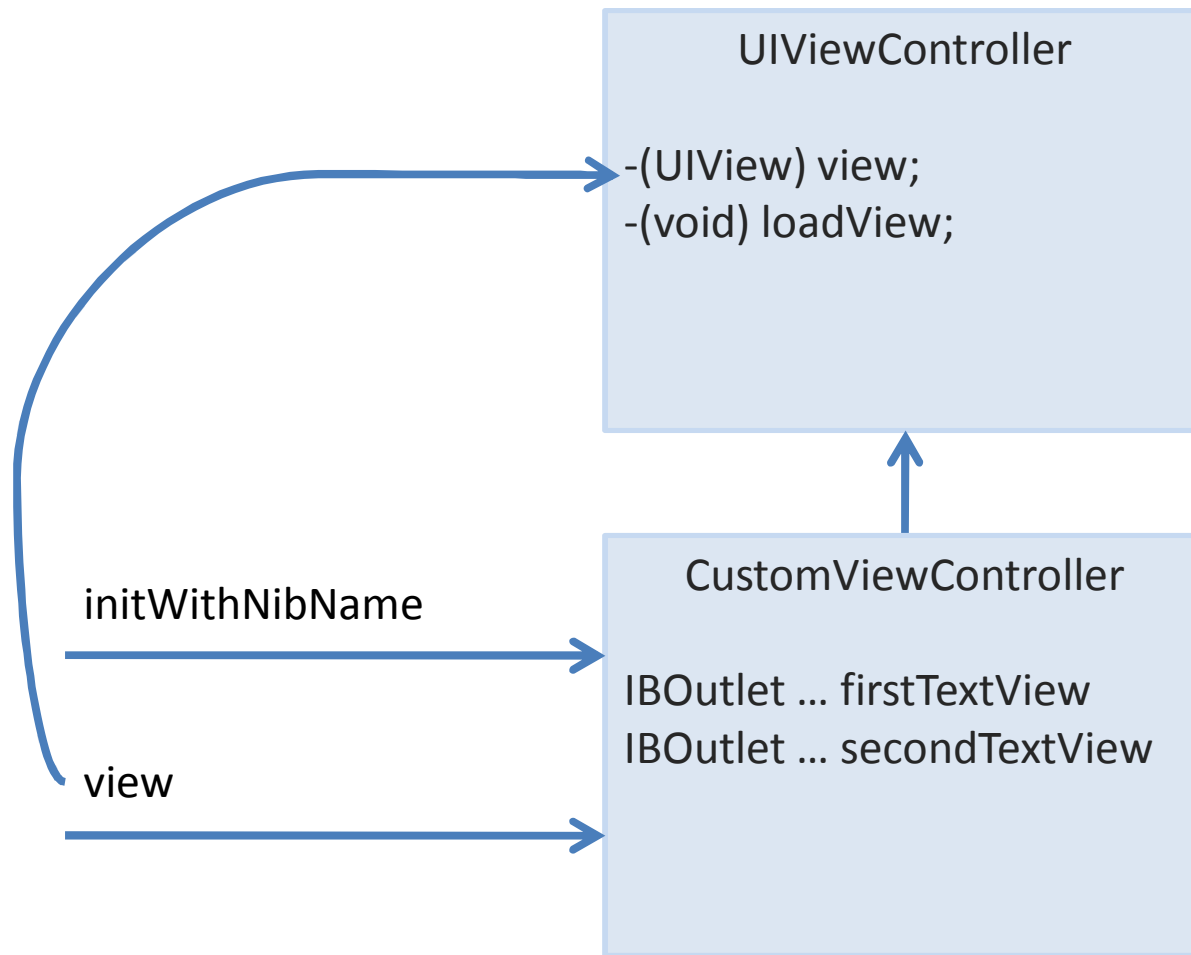
# loadView

---



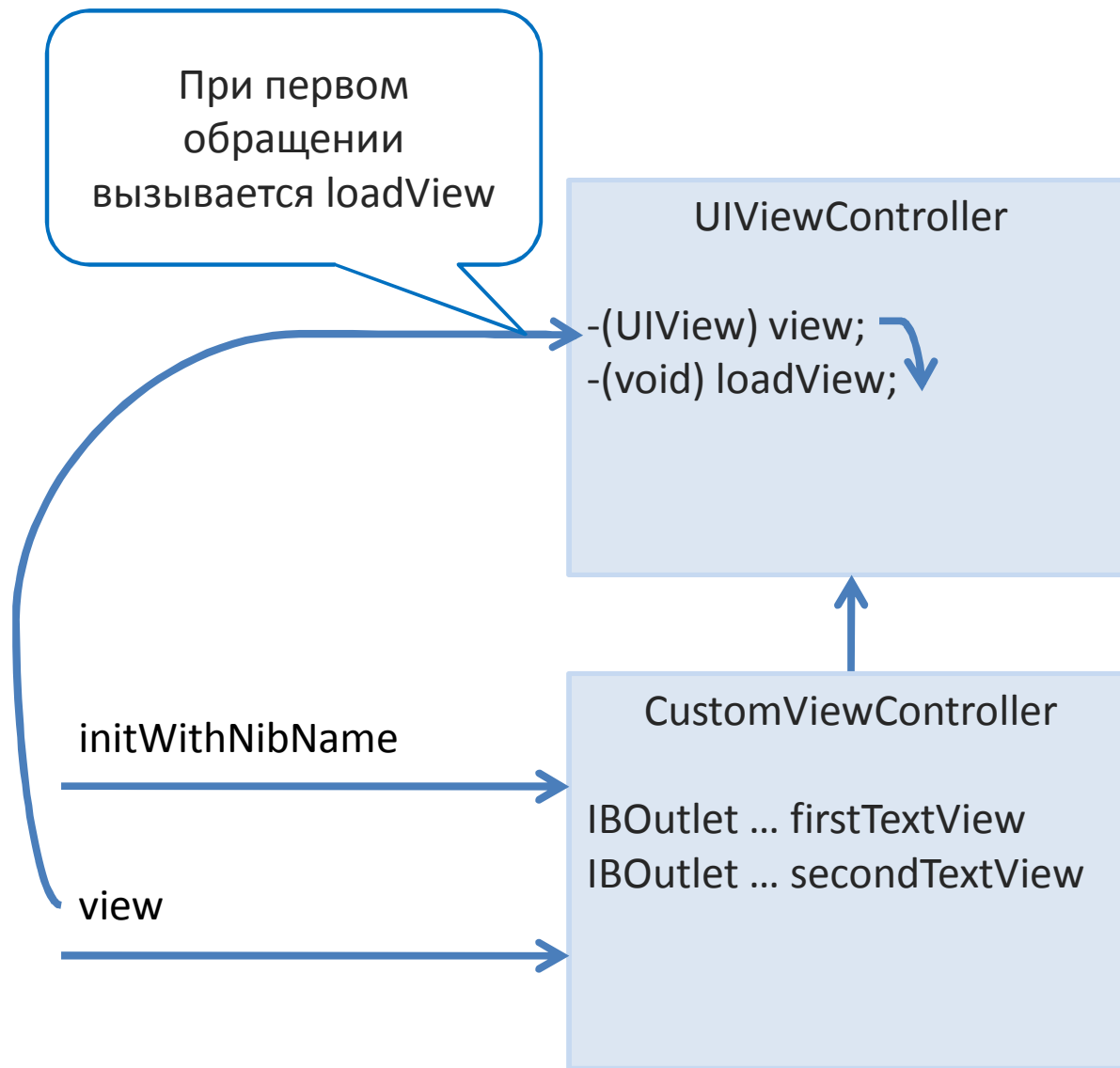
# loadView

---

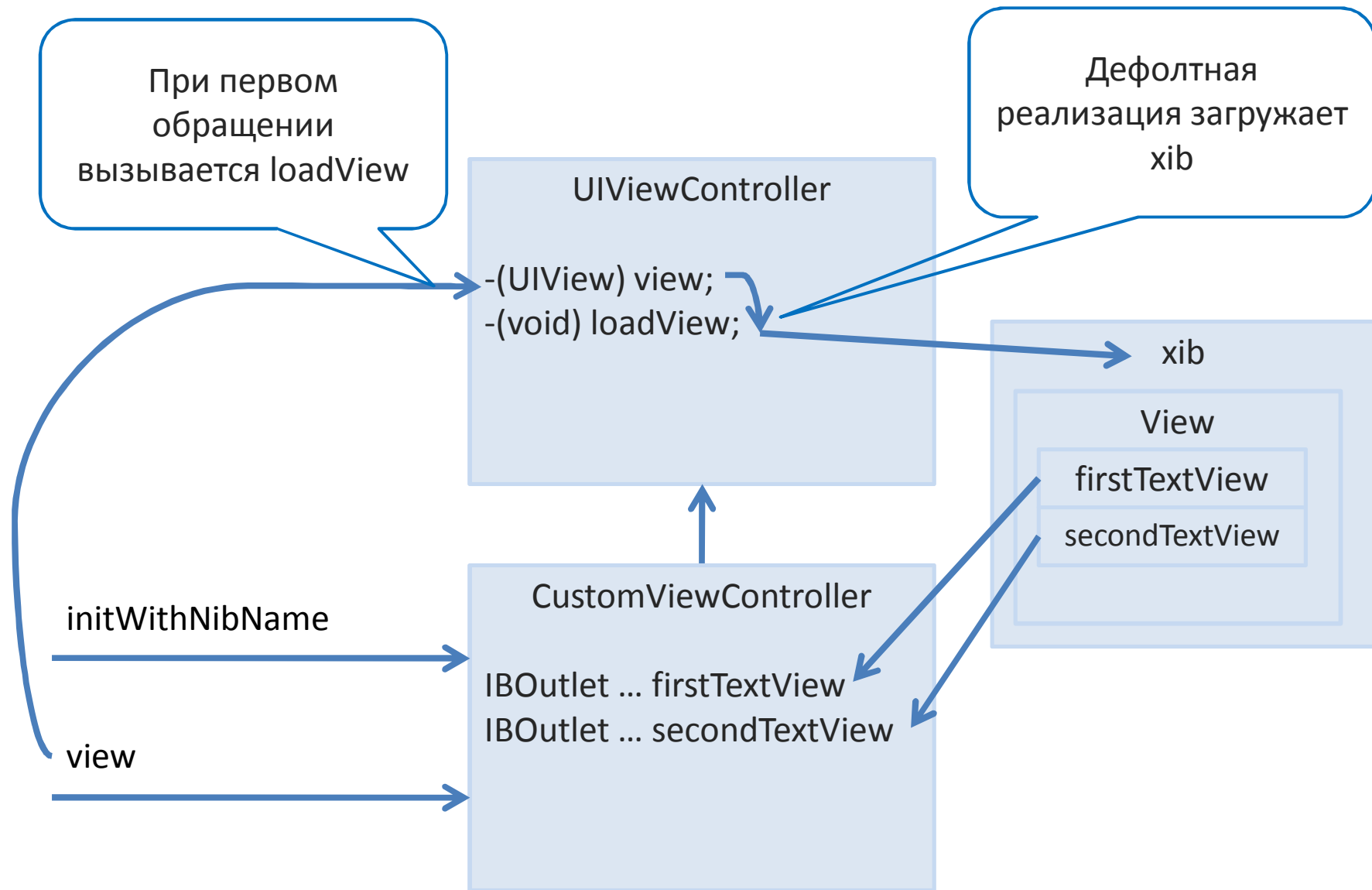


# loadView

---

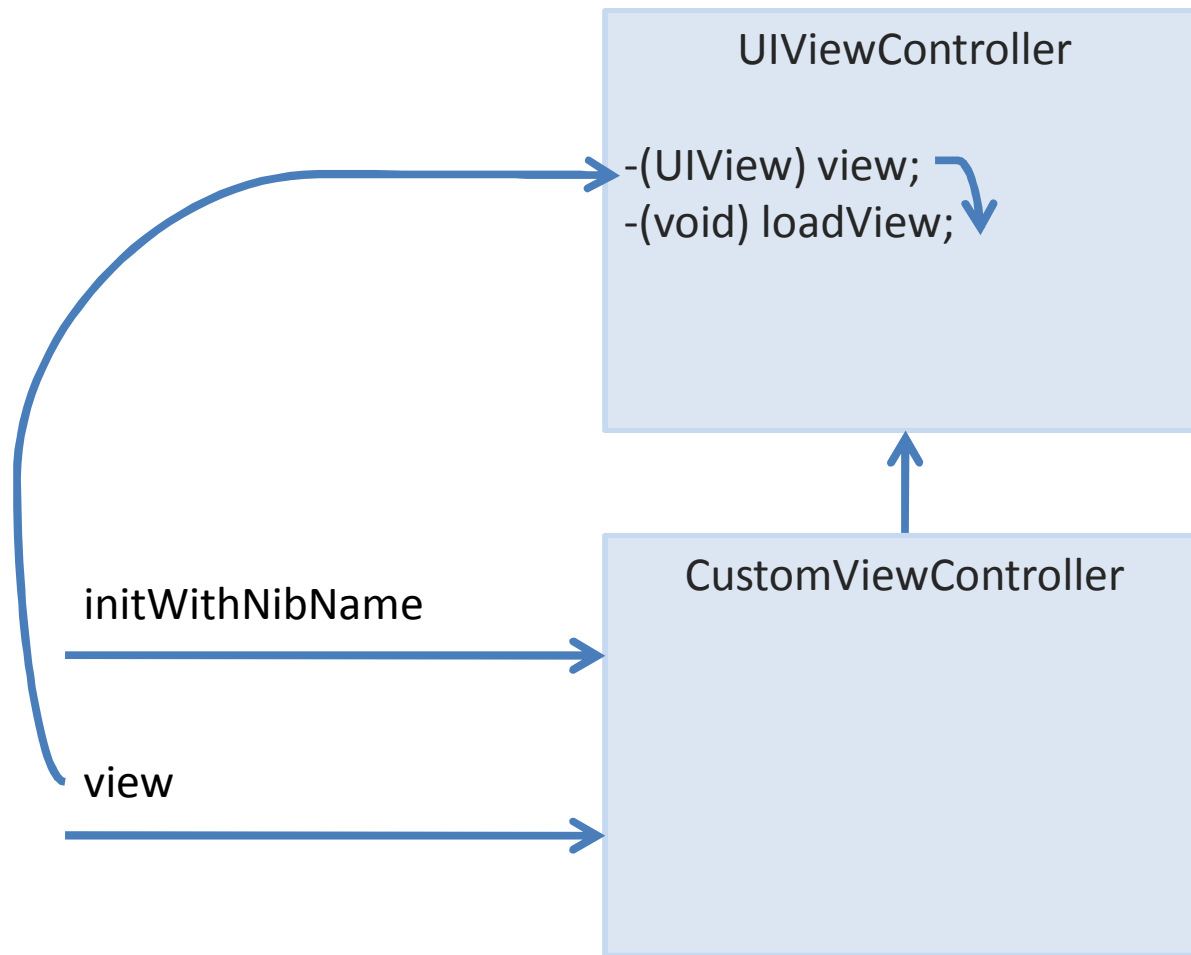


# loadView



# loadView

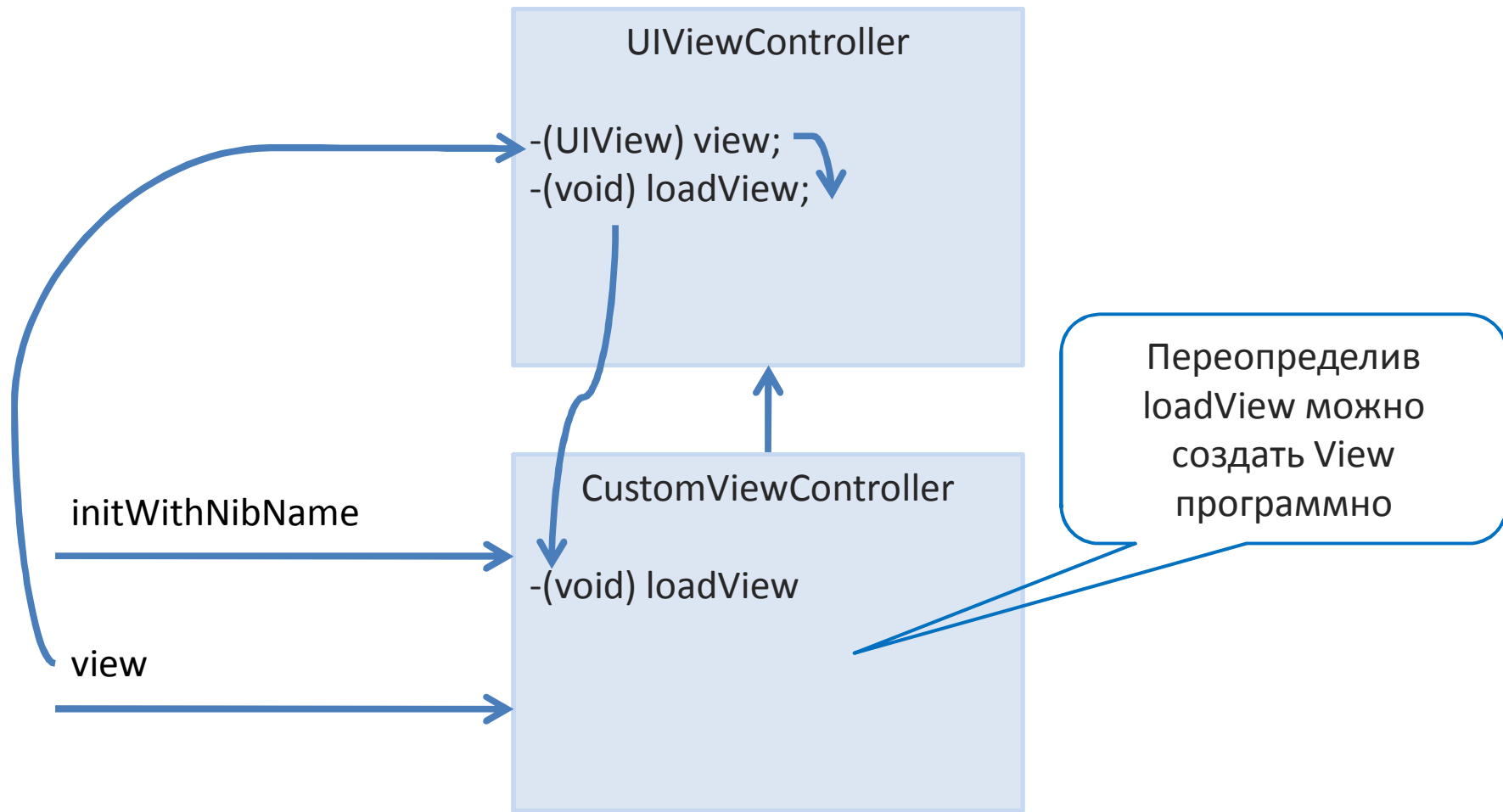
---





# loadView

---



# Как загружается xib

---

- `NSBundle`  
    `loadNibName:owner:`
- `[ [ NSBundle mainBundle ] loadNibNamed: @"MyView" owner: self ];`

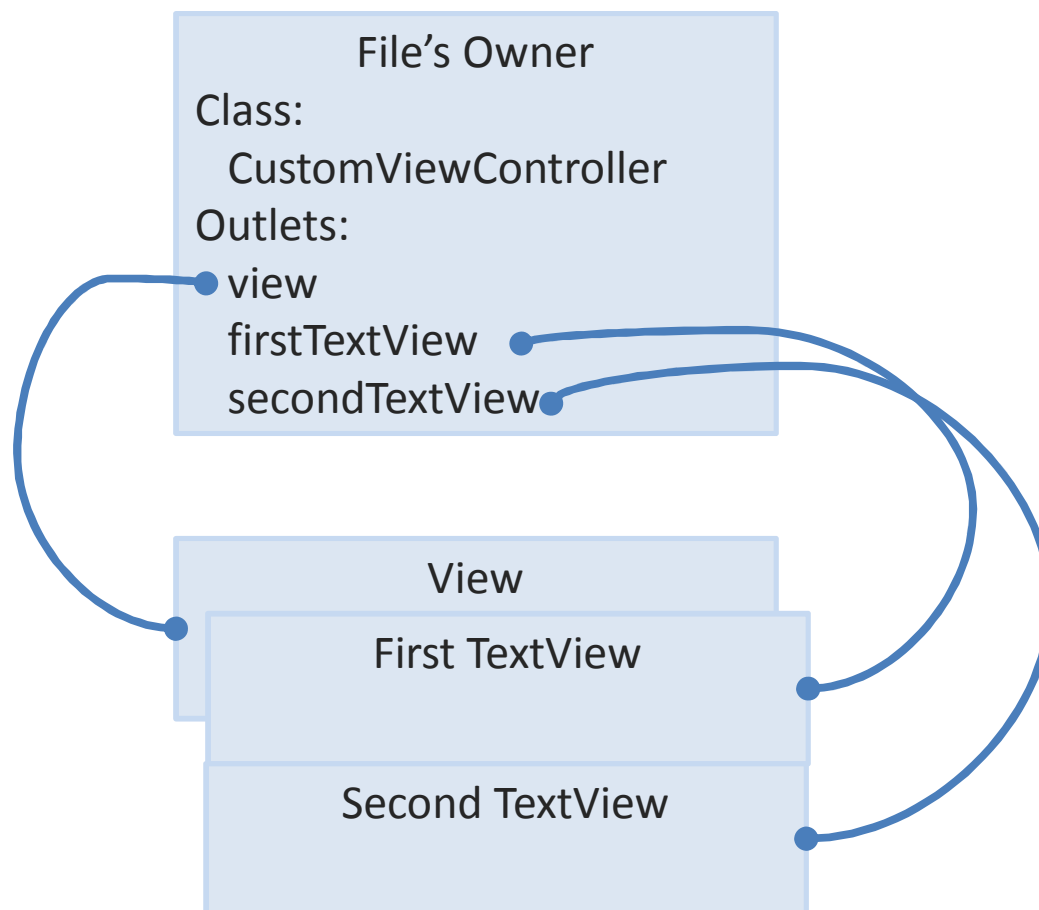
## Как загружается xib (в два этапа)

---

- Создание иерархии представлений (view) по xib
- Связывание Outlet'ов

# Простой пример

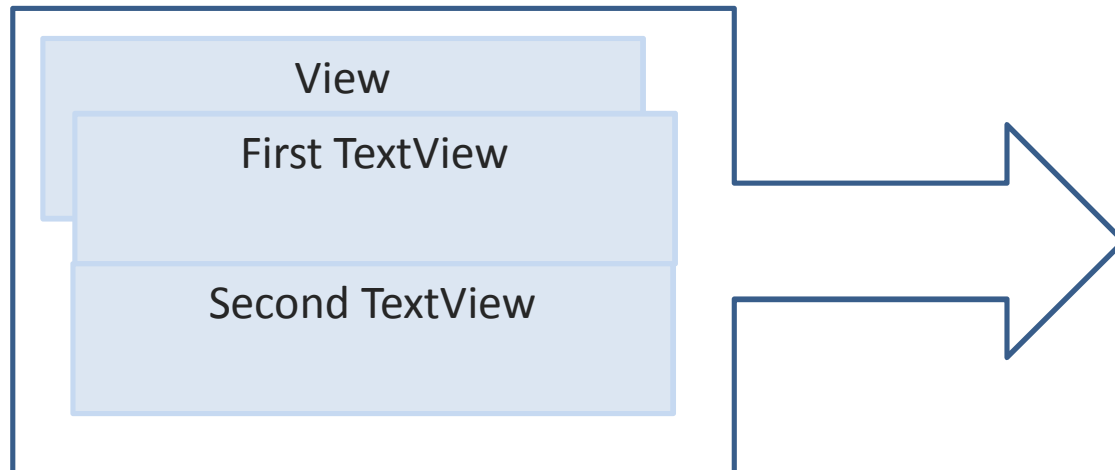
---



# Создание View

---

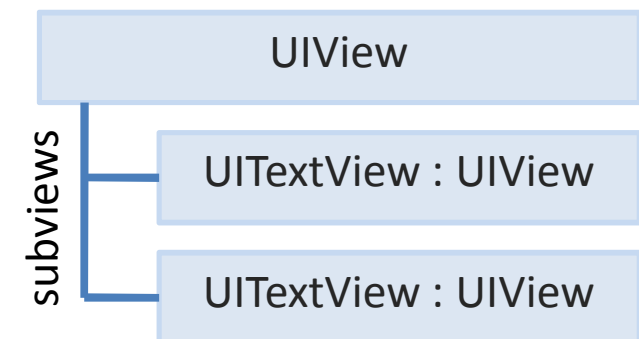
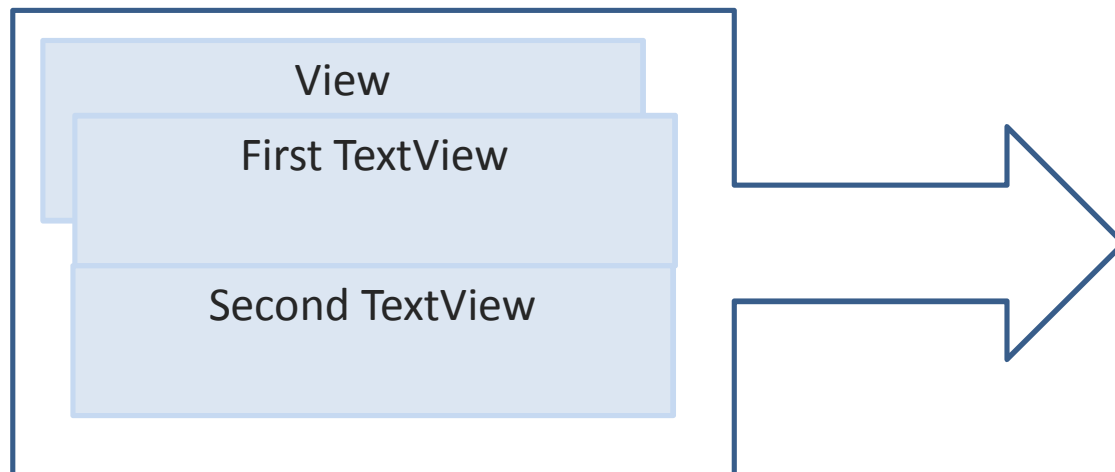
File's Owner  
Class:  
CustomViewController  
Outlets:  
view  
firstTextView  
secondTextView



# Создание View

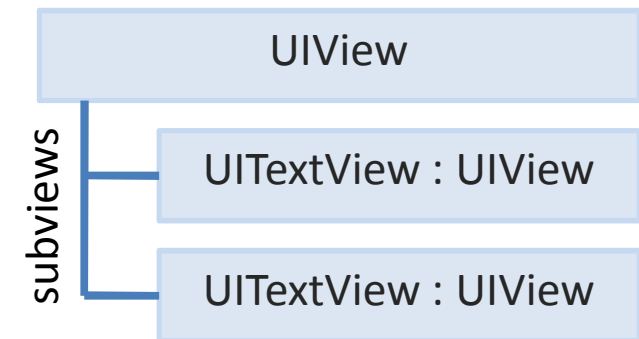
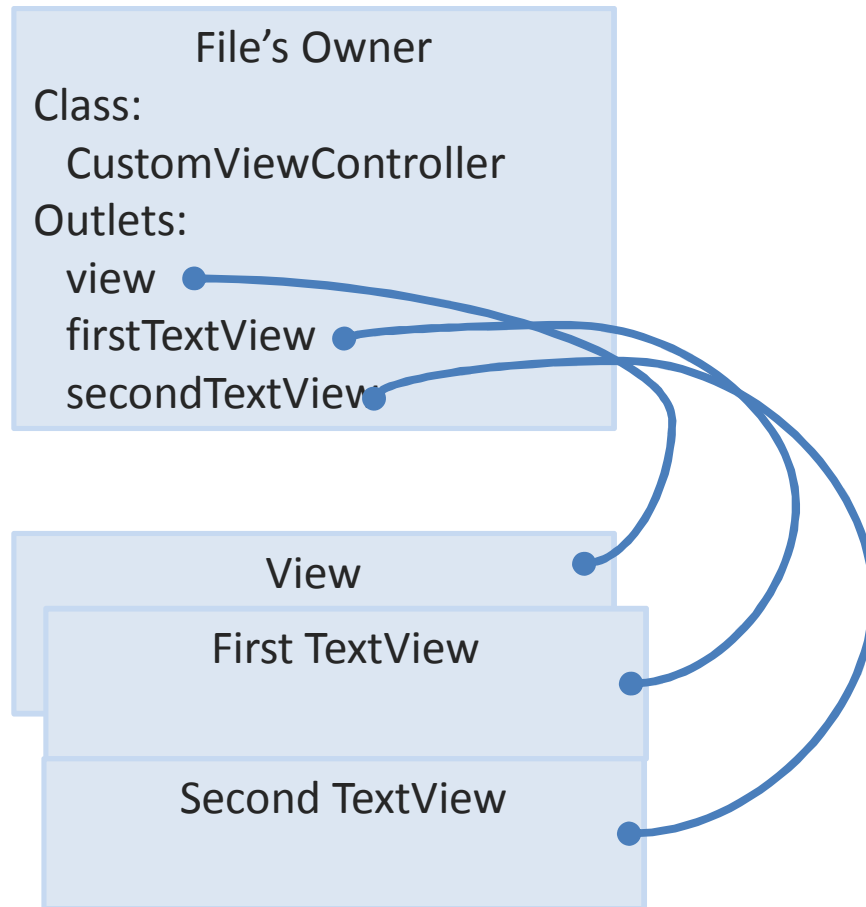
---

File's Owner  
Class:  
CustomViewController  
Outlets:  
view  
firstTextView  
secondTextView



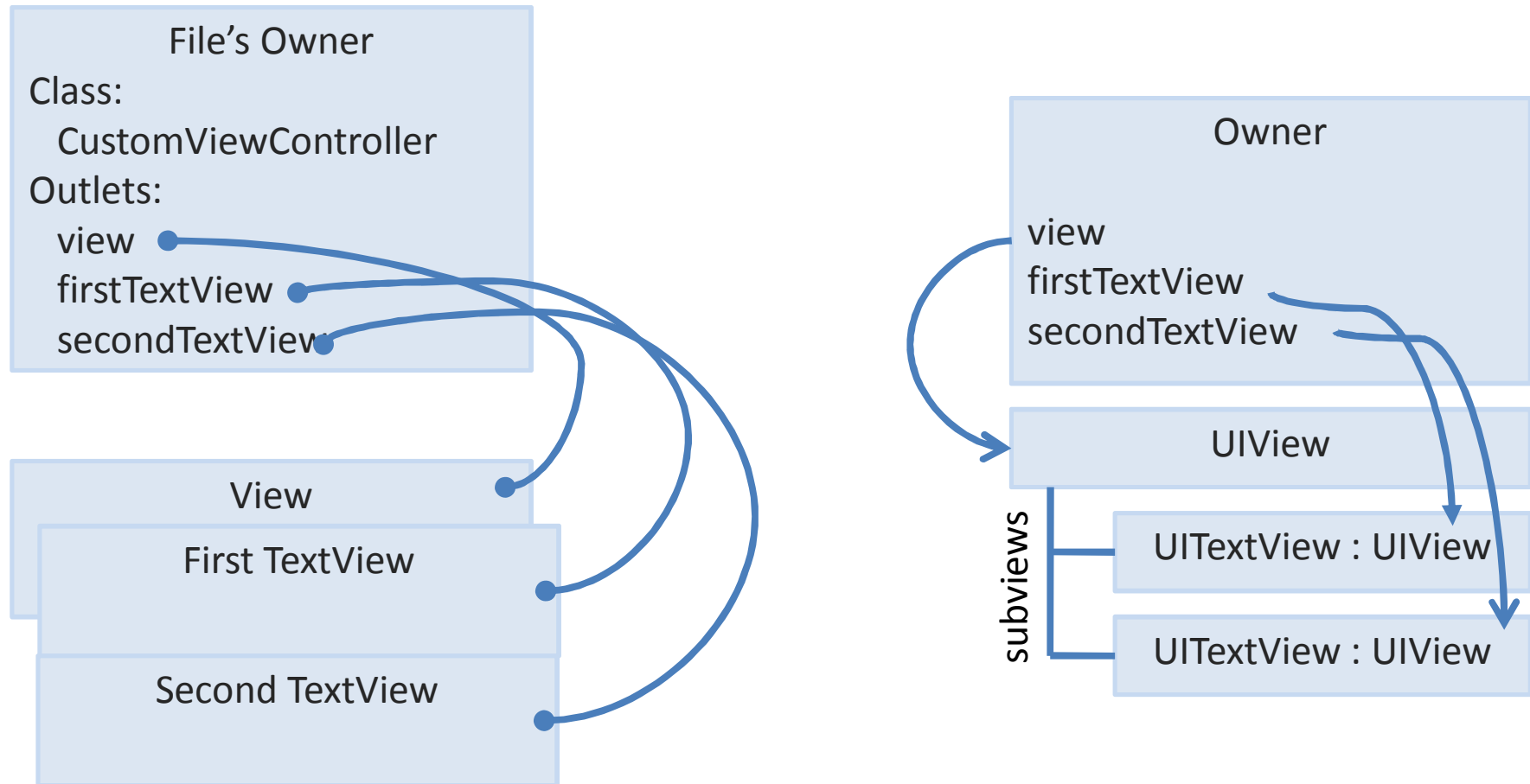
# Связывание

---



# Связывание

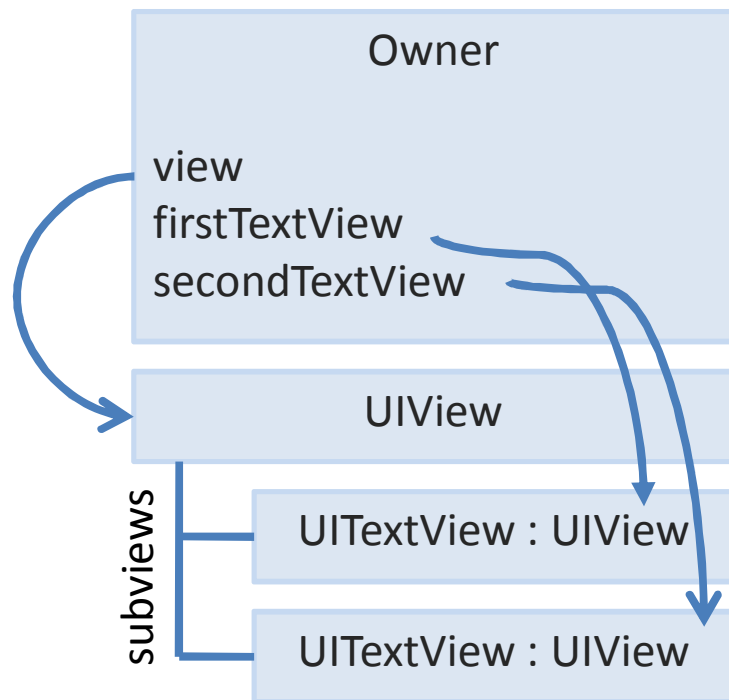
---





# Связывание

---



```
-(id) loadNibNamed: (NSString*) nib  
                  owner: (id) owner  
{  
    [ owner setFirstNameView: view1 ];  
    [ owner setSecondNameview: view2 ];  
}
```

# Делегат приложения (UIApplicationDelegate)

---

- application:didFinishLaunchingWithOptions
- applicationWillTerminate
- applicationWillResignActive
- applicationDidBecomeActive

# Делегат приложения (UIApplicationDelegate)

---

- application:didFinishLaunchingWithOptions

```
-( BOOL ) application: ( UIApplication* )application
    didFinishLaunchingWithOptions: ( NSDictionary* )options
{
    self.Window = [ [ [ UIWindow alloc ] initWithFrame: [ [ UIScreen
mainScreen ] bounds ] ] autorelease ];

    CustomViewController* ctrl = [ [ [ CustomViewController alloc ]
initWithNibName: @"CustomViewController" bundle: [ NSBundle
mainBundle ] ] autorelease ];

    self.Window.rootViewController = ctrl;
    [ self.Window makeKeyAndVisible ];

    return YES;
}
```

# Практическое задание

---

Практикуемся с UIView и UIViewController

1. Создать RssItemViewController для отображения информации об элементе RSS
2. В InterfaceBuilder сделать представление, для отображения
  1. Title
  2. Link
  3. Description
  4. ...
3. Воспользоваться UITextField и UILabel

См. Apple Guides

- View Controller Programming Guide for iOS
- View Programming Guide for iOS