iOS Lab for Dev

Developing Apps for iPhone and iPad

Lesson 2

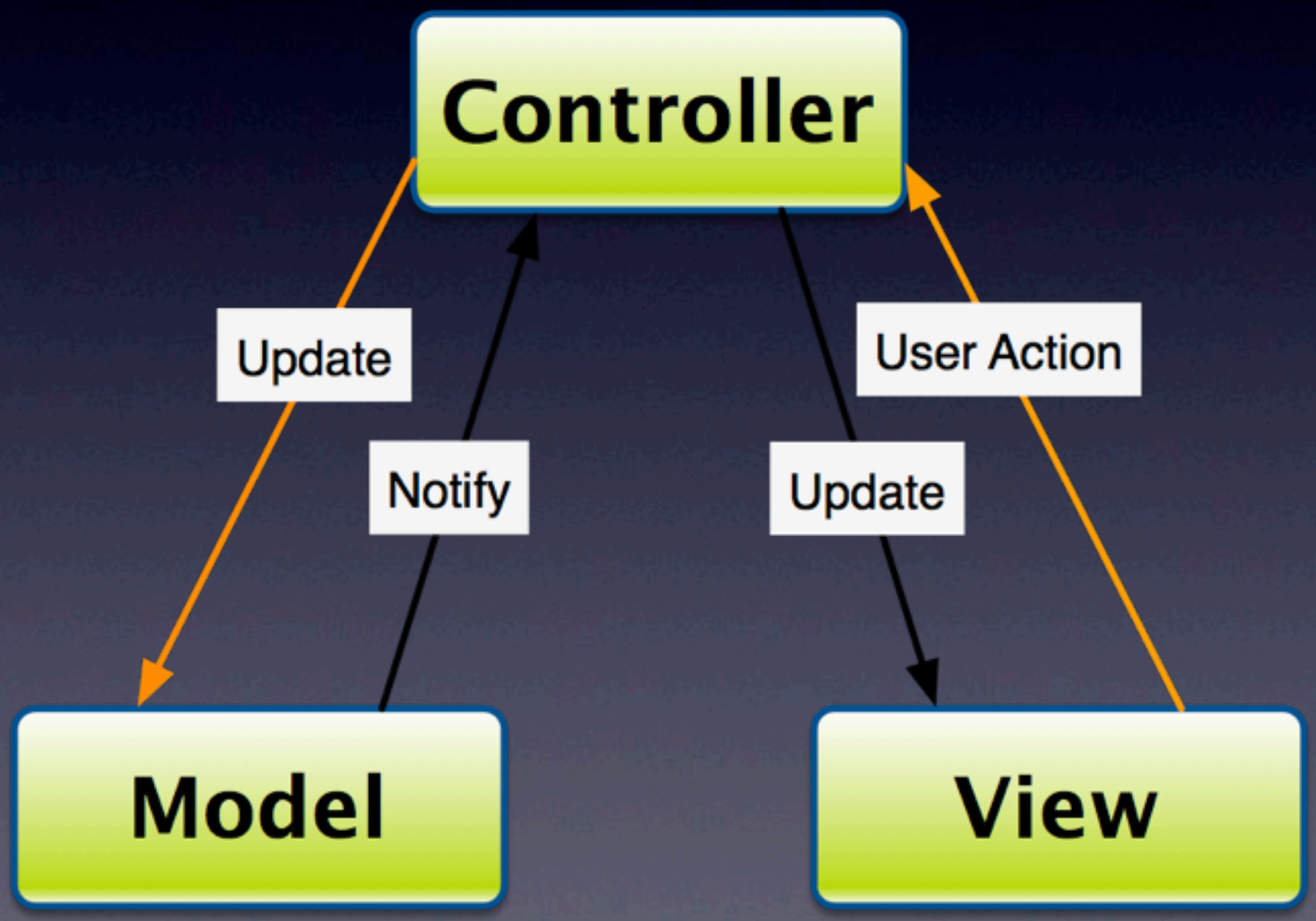
Topics of the day

Mattina
- MVC
- Delegation e UITableView

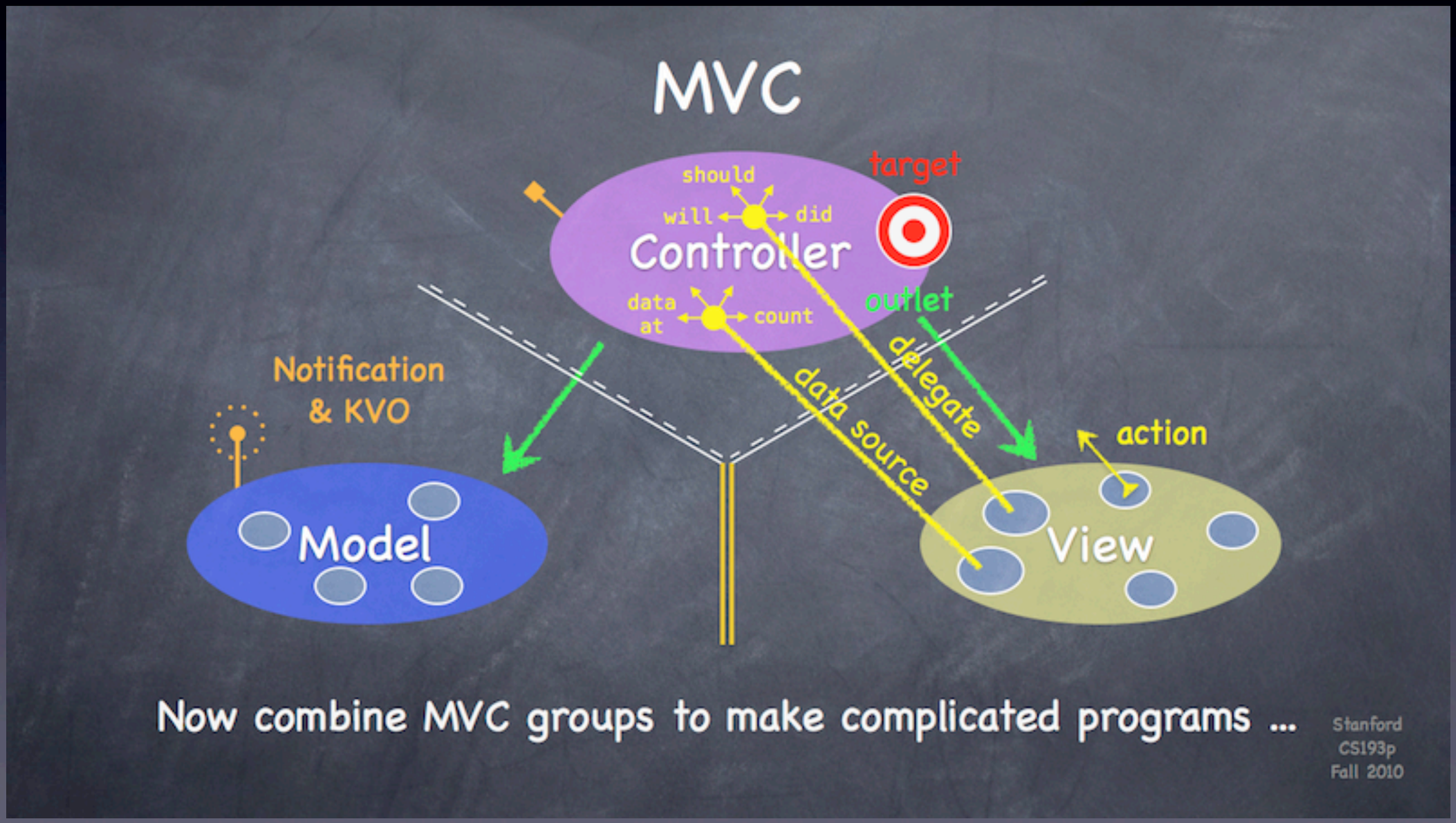
Pomeriggio
- UIView
- UIScrollview
- UIImageView

MVC
(Model-View-Controller)

MVC



MVC


MVC
should
will
did
Controller
data
at
count
target
outlet
delegate
data source
action
Notification
& KVO
Model
View
Now combine MVC groups to make complicated programs ...
Stanford
CS193p
Fall 2010

Delegation

Delegation

"In software engineering, the delegation pattern is a design pattern in object-oriented programming where an object, instead of performing one of its stated tasks, delegates that task to an associated helper object. There is an Inversion of Responsibility in which a helper object, known as a delegate, is given the responsibility to execute a task for the delegator. The delegation pattern is one of the fundamental abstraction patterns that underlie other software patterns such as composition (also referred to as aggregation), mixins and aspects."

http://en.wikipedia.org/wiki/Delegation_pattern

Delegation

I protocolli sono un meccanismo per definire dei metodi che una classe deve implementare.

L'interfaccia, in senso stretto, di una classe espone il suo repertorio di metodi pubblici; le entità esterne alla classe interagiscono con essa rispettando questo contratto.

Tanto quanto una classe espone metodi, così essa può anche rendere noto il fatto di invocare dei metodi facenti parte di un protocollo. Tali metodi devono essere implementati altrove e l'oggetto che *promette* di implementarli prende il nome di oggetto delegato.

La dichiarazione di un protocollo ha una sintassi simile alla seguente:

```
@protocol MioProtocollo
- (void)primoMetodoDaImplementare:(id)oggetto;
- (void)secondoMetodoDaImplementare:(id)oggetto;
@end
```

Delegation

La sintassi per dichiarare l'adozione di un protocollo nell'interfaccia e la successiva implementazione dei metodi è la seguente:

```
@interface MiaClasseDelegata <MioProtocollo> {
    // variabili di istanza ...
}
@end
```

```
@implementation MiaClasseDelegata
- (void)primoMetodoDaImplementare:(id)parametro { ... }
- (void)secondoMetodoDaImplementare:(id)parametro { ... }
@end
```

L'oggetto delegante che utilizza un protocollo per comunicare con entità esterne ad essa presenta un campo dati con la seguente sintassi:

```
id <MioProtocollo> delegate;
```

Delegation

Tale variabile di istanza viene settata con il riferimento all'oggetto delegato per rendere noto che sarà esso ad occuparsi dell'implementazione dei metodi del protocollo.

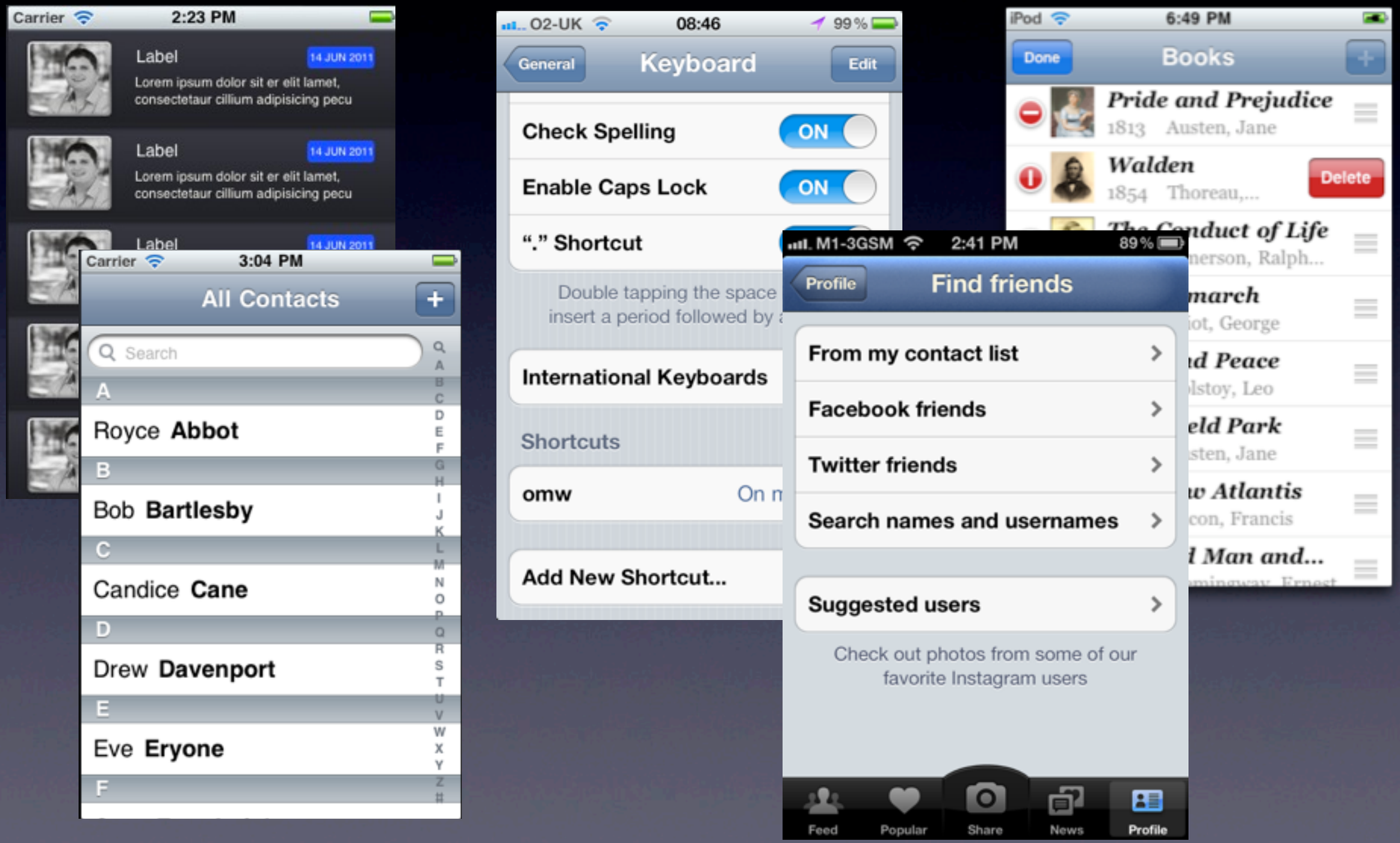
Quando l'oggetto delegante invoca i metodi che sono stati delegati, ciò viene fatto sull'oggetto delegato che ha tipo statico id in quanto non vi sono vincoli sul tipo dinamico della classe che ne fornisce l'implementazione.

```
[delegate primoMetodoDaImplementare:parametro];
```

Due principali svantaggi della delegazione sono la complicazione del codice nella lettura e l'eccessiva lunghezza che ne deriva dall'utilizzo di tale pattern. D'altra parte il delegation pattern è un pattern fondamentale dell'ingegneria del software che permette di esprimere molta potenza linguistica.

UITableView

UITableView



UITableViewDelegate

Configuring Rows for the Table View

1 – tableView:heightForRowAtIndexPath:
2 – tableView:indentationLevelForRowAtIndexPath:
3 – tableView:willDisplayCell:forRowAtIndexPath:

Managing Accessory Views

1 – tableView:accessoryButtonTappedForRowWithIndexPath:
2 – tableView:accessoryTypeForRowWithIndexPath: Deprecated in iOS 3.0

Managing Selections

1 – tableView:willSelectRowAtIndexPath:
2 – tableView:didSelectRowAtIndexPath:
3 – tableView:willDeselectRowAtIndexPath:
4 – tableView:didDeselectRowAtIndexPath:

Modifying the Header and Footer of Sections

1 – tableView:viewForHeaderInSection:
2 – tableView:viewForFooterInSection:
3 – tableView:heightForHeaderInSection:
4 – tableView:heightForFooterInSection:
…

- http://developer.apple.com/library/ios/#documentation/uikit/reference/UITableViewDelegate_Protocol/Reference/Reference.html

UITableViewDatasource

Configuring a Table View

```
1 – tableView:cellForRowAtIndexPath:  required method
2 – numberOfSectionsInTableView:
3 – tableView:numberOfRowsInSection:  required method
4 – sectionIndexTitlesForTableView:
5 – tableView:sectionForSectionIndexTitle:atIndex:
6 – tableView:titleForHeaderInSection:
7 – tableView:titleForFooterInSection:
```

Inserting or Deleting Table Rows

```
1 – tableView:commitEditingStyle:forRowAtIndexPath:
2 – tableView:canEditRowAtIndexPath:
```

Reordering Table Rows

```
1 – tableView:canMoveRowAtIndexPath:
2 – tableView:moveRowAtIndexPath:toIndexPath:
```

...

http://developer.apple.com/library/ios/#documentation/uikit/reference/UITableViewDataSource_Protocol/Reference/Reference.html

UIView, UIScrollView, UIImageView

UIView

Una view (istanza di UIView) rappresenta un’area rettangolare

E’ responsabile dell’interfaccia e degli eventi che accadono al suo interno

E’ la superclasse di moltissimi altri oggetti di UIKit, come ad esempio UILabel, UIImageView, UIScrollView, che quindi ereditano una serie di attributi e metodi.

Gerarchia di viste:
- ogni vista ha solo una superview
- ogni vista può avere un array di subviews, che vengono disegnate in ordine
- ogni istanza di UIViewController (e relative sottoclassi scritte da noi) ha la property view di tipo UIView (root delle gerarchia di view per lo specifico controller)
- UIWindow è la vista al top della gerarchia di view, ogni applicazione iOS ha una unica UIWindow

Concetti chiave

CGFloat

CGPoint (point, non pixel!) (C struct)

```
CGPoint p = CGPointMake(20.0, 30.0);
p.x;
p.y;
```

CGSize

```
CGSize s = CGSizeMake(200.0, 100.0);
s.width;
s.height;
```

CGRect

```
CGRect r = CGRectMake(20.0, 30.0, 200.0, 100.0);
r.origin.x; r.origin.y
r.size.width; r.size.height;
```

Coordinate

Tutte le coordinate sono relative all'origine (0.0, 0.0) che è nell'angolo in alto a SX

Ogni oggetto UIView ha tre proprietà:

```
CGRect bounds;
CGRect frame;
CGPoint center;
```

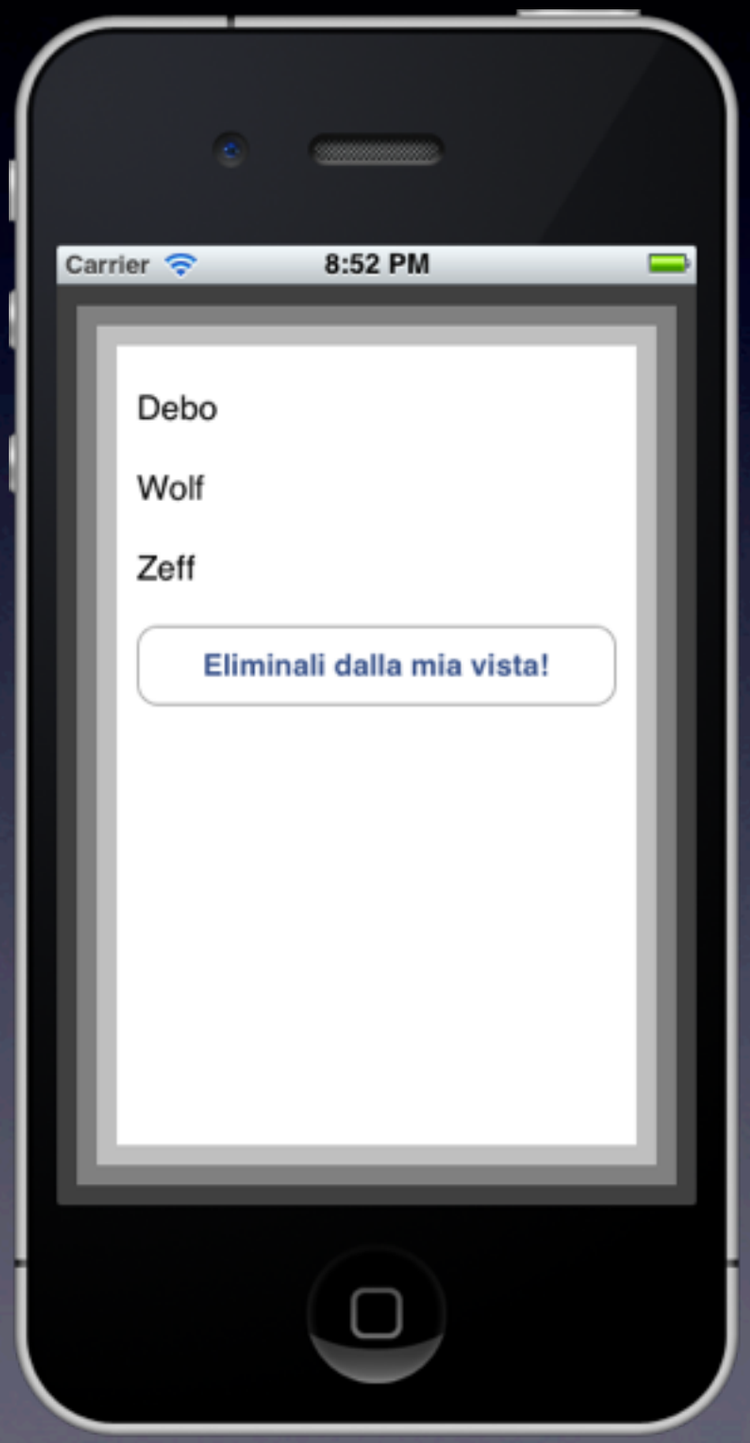
Frame: rettangolo della vista nelle coordinate della superview
Bounds: rettangolo della vista nelle coordinate della vista stessa (quasi sempre è (0.0, 0.0, width, height)
Center: punto del centro della vista, nelle coordinate della superview

Costruire UIView in XCODE

Demo vista dettaglio ingrediente:

- gerarchia (raggruppare logicamente elementi comuni)
- proprietà (esempio: vedere come UILabel eredita da UIView)
- posizionamento ed autoresizing mask
- tag
- sottoclassi custom di UIView

Costruire UIView via codice



M = Model

Gestisce, incapsula i dati dell'applicazione.

Varie possibilità in iOS, in ordine di complessità:
- UserDefaults: dizionario persistito dal sistema operativo
- Plist: dizionari o array salvati su file sul filesystem della sandbox
- CoreData: oggetti persistenti salvati su DB Sqlite
- DB Sqlite: uguale a SQLite usato server side o su desktop
- API Rest: il server salva tutti i dati, l'app interroga delle viste in JSON

Strategia: singleton. Per i nostri esempi useremo una variabili di istanza dell'application delegate (che è instanziato per noi una ed una sola volta)

UIScrollView

UIScrollView

Da usare tutte le volte che il contenuto da visualizzare è più grande della vista

Proprietà:

CGRect contentSize

CGPoint contentOffset

Costruire UIScrollView programmaticamente

Esercizio complesso: WebBrowser

Creare una componente DAWebBrowserViewController web browser (View Controller) con le funzionalità base di navigazione (caricamento URL, refresh, stop).

Scrivere un protocollo DAWebBrowserViewControllerDelegate per tale componente con almeno 2 metodi delegati.

Nella componente utilizzare i protocolli UITextFieldDelegate (per barra degli indirizzi) e UIWebViewDelegate (per la webview).

Durante il caricamento mostrare un HUD sulla webview utilizzando la componente open source MBProgressHUD disponibile su GitHub.

Punti extra:

gestire la history della navigazione tramite un metodo di DAWebBrowserViewControllerDelegate

N.B. Esercizio svolto disponibile sul repository del corso