#7 Talk to me baby

Make our classes talk to each other

Part 1 – Protocols

And delegates, this time into detail

What is a protocol?

- An interface (sort of)
  - Protocols are actually much more versatile, which is simultaneously a PITA and a blessing
  - Unlike interfaces, they can be defined almost anywhere, and implemented almost anywhere
- Other than defining a list of methods another class has to implement, they are also commonly used to transfer data between view controllers

What do you need to create a working protocol?

1. A protocol declaration

```
@protocol NewCommentDelegate <NSObject>

- (void)reloadComments;

@end
```

2. A protocol implementation

```
@interface PostDetailsViewController : UIViewController
<NewCommentDelegate>
```

3. A delegate

```
@property (nonatomic, weak) id<NewCommentProtocol> newCommentDelegate;
```

When to use a protocol?

- When you want to enforce implementation of certain methods in other classes that use code from other classes
- When you want to implement a one-to-one communication between view controllers or classes
- When you want to leave the implementation anonymous (or you don’t know which class will be using certain resources)
- To conform to a certain development architecture

Demo

Part 2 – Notifications

Not the standard kind the users see

NSNotificationCenter

- http://nshipster.com/nsnotification-and-nsnotificationcenter/
- A crossroad class for notification in your application
- Somewhat similar to the publish-subscribe pattern
- NSNotification is the notification object being passed

How to use NSNotificationCenter

1. Publish a notification

```
[[NSNotificationCenter defaultCenter] postNotificationName:@"NewComment"
object:nil];
```

2. Receive a notification somewhere else

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(reloadComments) name:@"NewComment" object:nil];
```

3. Remove the observer as soon as you don’t need it anymore

```
- (void)dealloc
{
    [[NSNotificationCenter defaultCenter] removeObserver:self];
}
```

When to use NotificationCenter

- **Almost** never yourself :)
- When communication goes from one to many
- When listening for system notifications (keyboard showing up, movie player notifications etc.)

Demo

How to decide?

- http://www.objc.io/issues/7-foundation/communication-patterns/

DZ 6

Make ‘em talk

Komentiranje threadova i refresh

Potrebno je na prethodnu zadaću do kraja implementirati novi kontroler koji služi za dodavanje komentara na postojeći post. Nakon što se komentar uspješno posta, potrebno je automatski dismissati kontroller za komentiranje i refreshati komentare na ekranu koji pokazuje detalje posta. Upotreba protokola ili notifikacija po volji :).

DZ za grindere

Tijekom paginacije, obično je najlakše kod updatea dataSourcea jednostavno pozvati reloadData nad tableViewom. Međutim, ovo je moguće izvesti puno elegantnije i korisnicima vizualno upečatljivije ako se koriste metode za animirano umetanje cellova.

Keywords: beginUpdates, endUpdates nad tableViewom.

Appendix

- **Protocols**
  - https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/WorkingwithProtocols/WorkingwithProtocols.html
- **NSNotificationCenter**
  - http://nshipster.com/nsnotification-and-nsnotificationcenter/
  - https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSNotificationCenter_Class/
- **Further reading**
  - http://www.objc.io/issues/7-foundation/communication-patterns/