Delegation

Lecture 9
Delegation?
Delegation?

- **delegate**: verb /ˈdeləˌɡæt/
  Send or authorize (someone) to do something as a representative
Delegation?

- From Apple's Cocoa Core Competencies:

"Delegation is a simple and powerful pattern in which one object in a program acts on behalf of, or in coordination with, another object."
Delegation, Defined

*This slide taken directly from Apple docs*
Delegation, Defined

- The *delegating* object keeps a reference to another object - the *delegate* - and at the appropriate time sends a message to it.

- The message informs the delegate of an event that the delegating object is about to handle or has just handled.

*This slide taken directly from Apple docs*
Delegation, Defined
Delegation, Defined

• "The delegate may respond to the message by updating the appearance or state of itself or other objects in the application, and in some cases it can return a value that affects how an impending event is handled."

*This slide taken directly from Apple docs*
Delegation Example
Delegation
Delegation

- Delegation is actually a very simple idea
Delegation

• Delegation is actually a very simple idea
• It allows for powerful customization of the behavior of objects and the applications that use them
Delegation

- Delegation is actually a very simple idea
- It allows for powerful customization of the behavior of objects and the applications that use them
- Cocoa and Cocoa Touch define over 100 protocols named `SomethingDelegate`
  - Many belong to AppKit/UIKit
Delegation is best learned by example!
NSAppDelegate
NSAppDelegate

- (void) applicationDidFinishLaunching:
  (NSNotification *) aNotification;

• Commonly used by the delegate to perform additional initialization
• Called after the main run loop has been created, but before it gets any events
NSApplicationDelegate

- (NSApplicationTerminateReply) applicationShouldTerminate:
  (NSApplication *) sender

  • Called after the Quit menu item has been selected
  • Generally, you want to terminate immediately
    • But you can cancel termination or delay it
NSApplicationDelegate

- (void) applicationWillTerminate:
  (NSNotification *) aNotification;

• Delegates use this method to perform last-minute cleanup before the application terminates
NSAppDelegate

- (void) applicationWillBecomeActive: (NSNotification *) aNotification;
- (void) applicationDidBecomeActive: (NSNotification *) aNotification;
- (void) applicationWillResignActive: (NSNotification *) aNotification;
- (void) applicationDidResignActive: (NSNotification *) aNotification;

• The application will call these methods on the delegate as part of the application's gaining or losing active status
• All the information about these events is encapsulated by the NSNotification
  • In particular, the application in question
NSApplicationDelegate

- (void) applicationWillHide: (NSNotification *) aNotification;
- (void) applicationDidHide: (NSNotification *) aNotification;
- (void) applicationWillUnhide: (NSNotification *) aNotification;
- (void) applicationDidUnhide: (NSNotification *) aNotification;

• The application will call these methods on the delegate as part of the hiding and unhiding processes
NSAppDelegate

- `(NSMenu *) applicationDockMenu:
  (NSApplication *) sender;

• Use this method to dynamically provide a menu for the Dock icon (which appears when the user secondary-clicks the Dock icon)
NSAppDelegate

- (BOOL) application: (NSApplication *) theApplication
  openFile: (NSString *) filename;
- (BOOL) applicationOpenUntitledFile: (NSApplication *) theApplication;

• Open an existing file or create a new file
• Boolean return value allows the method to indicate whether or not the delegate successfully opened/created the file
Demo
NSWindowDelegate
**NSWindowDelegate**

- `(NSSize) windowWillResize: (NSWindow *) sender
toSize: (NSSize) frameSize;
- `(void) windowDidResize: (NSNotification *) notification;

• These methods will be called during resize operations (whether initiated by the user or programmatically)

typedef struct _NSSize {
  CGFloat width;    /* should never be negative */
  CGFloat height;   /* should never be negative */
} NSSize;
NSWindowDelegate

- (void) windowWillMiniaturize: (NSNotification *) notification;
- (void) windowDidMiniaturize: (NSNotification *) notification;
- (void) windowDidDeminiaturize: (NSNotification *) notification;

• These methods will be called during the minimizing and de-minimizing processes
**NSWindowDelegate**

- *(void)* windowWillEnterFullscreen: *(NSNotification *)* notification;
- *(void)* windowDidEnterFullscreen: *(NSNotification *)* notification;
- *(void)* windowWillExitFullscreen: *(NSNotification *)* notification;
- *(void)* windowDidExitFullscreen: *(NSNotification *)* notification;

- New with OS X Lion (10.7)!
- These methods are called when activating/deactivating full screen mode
- Use these methods to rearrange your views as needed to take advantage of full screen
NSWindowDelegate

- (void) windowWillMove: (NSNotification *) notification;
- (void) windowDidMove: (NSNotification *) notification;
- (void) windowDidChangeScreen: (NSNotification *) notification;

• These methods are called while moving windows around.
• If the user has multiple displays connected, windowDidChangeScreen: is called when the window is moved between displays.
NSWindowDelegate

- (BOOL) windowShouldClose: (id) sender;
- (void) windowWillClose: (NSNotification *) notification;

• These methods allow customization of window closing behavior
• Note that -windowShouldClose: will NOT be called when the user quits
  • Only when the user attempts to close a window explicitly
Demo
NSURLConnectionDelegate
First, a little motivation
First, a little motivation

• Apps frequently need to get data from the Web
First, a little motivation

• Apps frequently need to get data from the Web

• Often using HTTP requests
First, a little motivation

- Apps frequently need to get data from the Web
- Often using HTTP requests
- These requests are usually slow
First, a little motivation

• Apps frequently need to get data from the Web
  • Often using HTTP requests
• These requests are usually slow
  • But apps should not block the main thread while making these requests
First, a little motivation

- Apps frequently need to get data from the Web
  - Often using HTTP requests
- These requests are usually slow
  - But apps should not block the main thread while making these requests
- An unresponsive UI is a **BAD UI**
First, a little motivation
First, a little motivation

- Solution: make these requests "asynchronous"
First, a little motivation

• Solution: make these requests "asynchronous"

• Create an **NSURLConnection** object and give it a request (**NSURLRequest**)
First, a little motivation

- Solution: make these requests "asynchronous"

- Create an `NSURLConnection` object and give it a request (`NSURLRequest`)

- The `NSURLConnection` can then run the request on a background thread and message its delegate when interesting events happen
NSURLConnectionDelegate

- (void) connection: (NSURLConnection *) connection
didReceiveResponse: (NSURLResponse *) response;

• Called "when the connection has received sufficient data to construct the URL response for its request"
**NSURLConnectionDelegate**

- (void) connection: (NSURLConnection *) connection
didReceiveData: (NSData *) data;

• The connection incrementally calls this as it receives data
• The delegate should be prepared to concatenate several (many) of these together to build up the complete data response
NSURLConnectionDelegate

- (void)connectionDidFinishLoading: (NSURLConnection *)connection;
- (void)connection: (NSURLConnection *)
didFailWithError: (NSError *)error;

• Called when the connection finishes or encounters an error
• Exactly one of these will be called
  • After which, the delegate will receive no further messages
Demo
NSXMLParserDelegate
NSXMLParser
NSXMLParser

- Foundation provides NSXMLParser, a really great class for parsing XML documents
NSXMLParser

- Foundation provides **NSXMLParser**, a really great class for parsing XML documents
- Operates in an "event-driven" manner
NSXMLParser

• Foundation provides **NSXMLParser**, a really great class for parsing XML documents

• Operates in an "event-driven" manner

• **Does NOT build up a tree of the XML document**
NSXMLParser

• Foundation provides **NSXMLParser**, a really great class for parsing XML documents

• Operates in an "event-driven" manner

  • Does NOT build up a tree of the XML document

  • Rather, it messages its delegate each time it encounters some part of the XML document
NSXMLParserDelegate

- (void)parserDidStartDocument: (NSXMLParser *)parser;
- (void)parserDidEndElement: (NSXMLParser *)parser;

• Called when the parser begins and ends parsing the XML document
NSXMLParserDelegate

- (void) parser: (NSXMLParser *) parser
didStartElement: (NSString *) elementName
    namespaceURI: (NSString *) namespaceURI
    qualifiedName: (NSString *) qualifiedName
    attributes: (NSDictionary *) attributeDict;

• This message is sent when the parser encounters an opening tag
• Passes as argument:
  • The element name
  • Namespace information (if enabled)
  • The attribute key-value pairs
NSXMLParserDelegate

- (void) parser: (NSXMLParser *) parser
didEndElement: (NSString *) elementName
namespaceURI: (NSString *) namespaceURI
qualifiedName: (NSString *) qualifiedName;

• This message is sent when the parser encounters a closing tag
• Passes as argument:
  • The element name
  • Namespace information (if enabled)
NSXMLParserDelegate

- (void) parser: (NSXMLParser *) parser
  foundCharacters: (NSString *) string;

• Called when the parser encounters a sequence of characters not within a tag
• Greedy - it will send as many characters as possible
Demo
Summary

• Delegation is a simple yet powerful design pattern

• Provides an alternative to subclassing

• Allows a single object to customize the appearance and behavior of many objects

• Single source of truth!