IOS Design Patterns

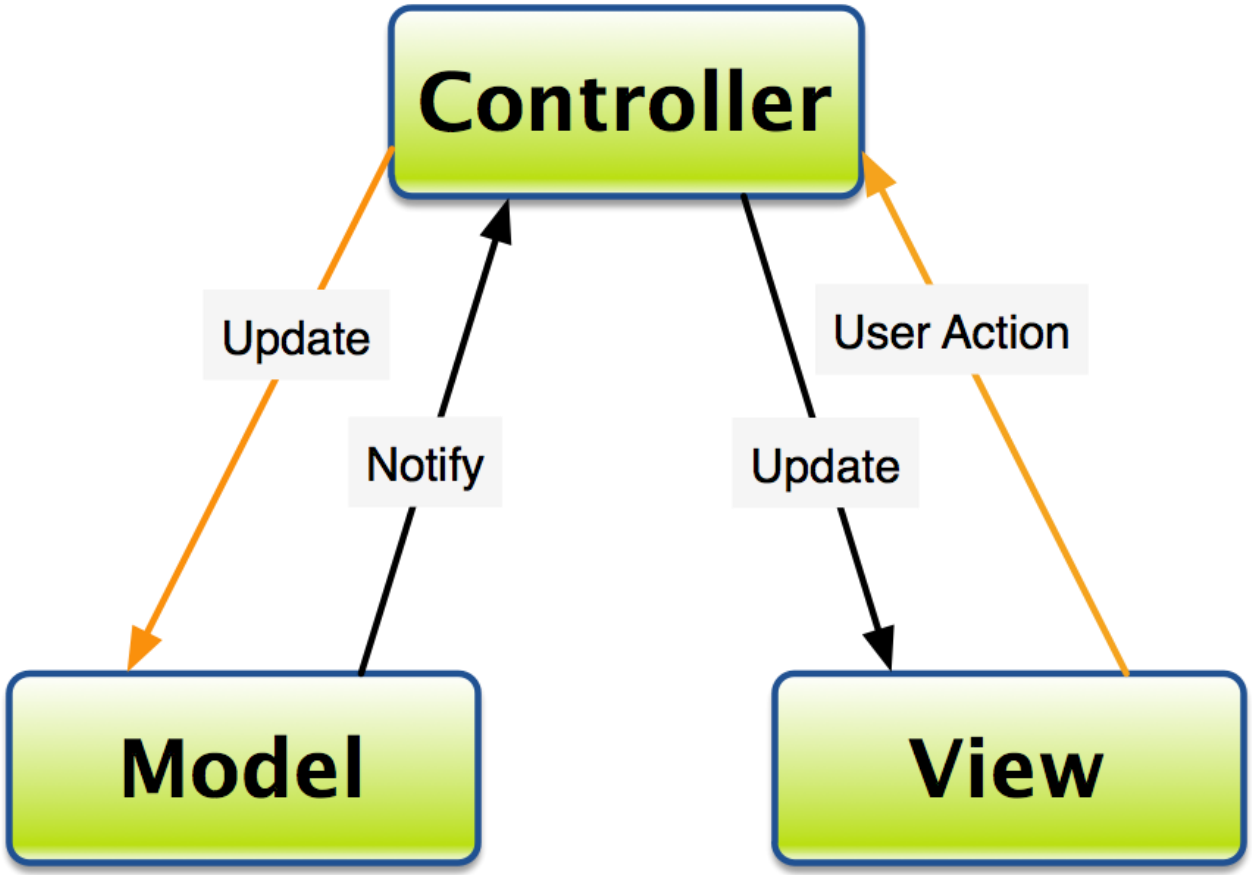
MVC, Singleton, Delegation, Target-action, Facade

Sisoft Technologies Pvt Ltd
SRC E7, Shipra Riviera Bazar, Gyan Khand-3, Indirapuram, Ghaziabad
Website: www.sisoft.in Email:info@sisoft.in
Phone: +91-9999-283-283

What Is a Design Pattern?

- A design pattern is a template for a design that solves a general, recurring problem in a particular context
- It is a tool of abstraction that is useful in fields like architecture and engineering as well as software development

MVC Communication pattern



View objects

- are visible to the user
- the buttons, labels, and the view they are placed on top of are all view objects
- Views are usually standard UIView subclasses (UIButton, UISlider), but you will sometimes write custom view classes

Model objects

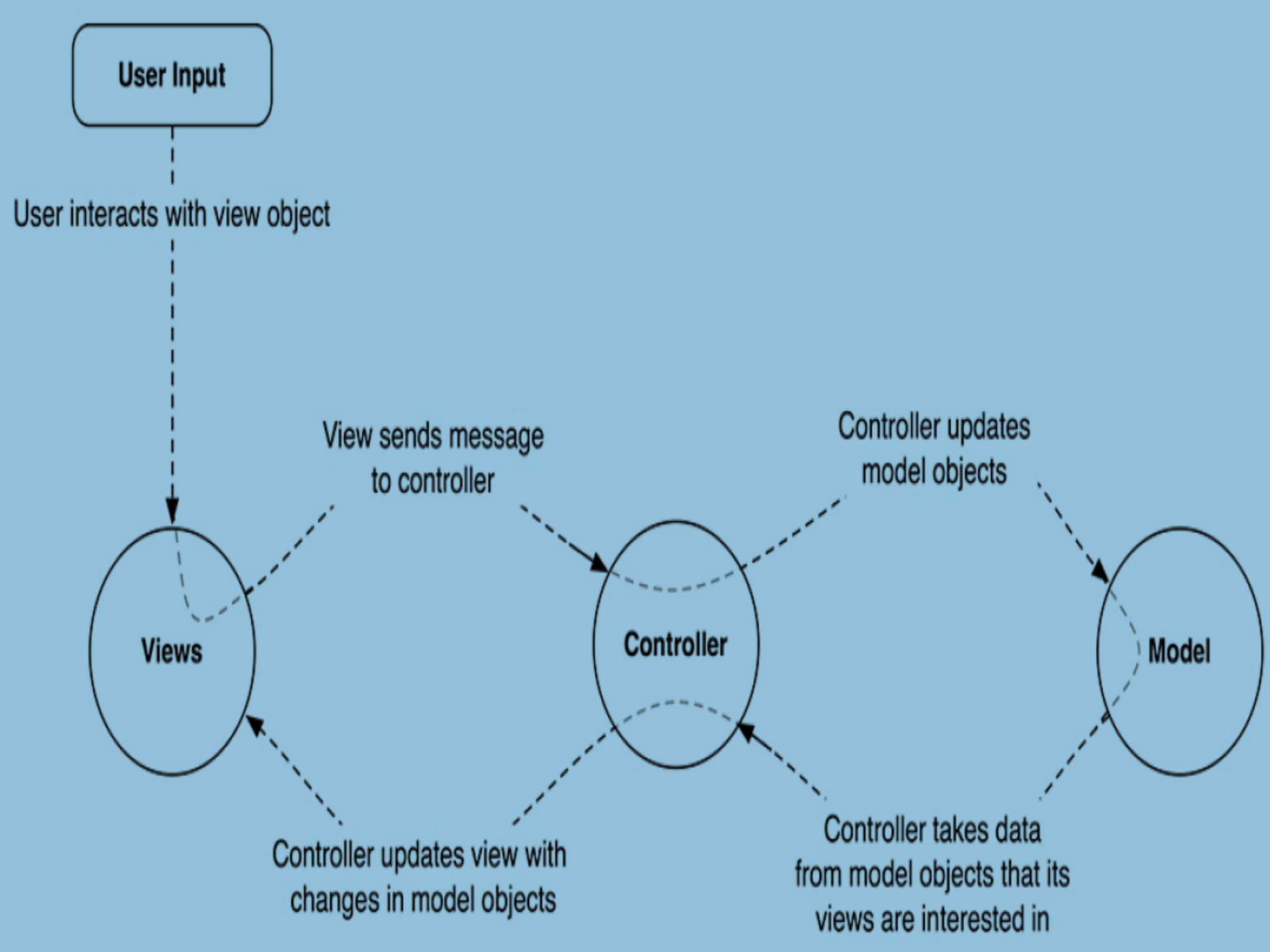
- Hold data and know nothing about the user interface
- Model objects typically use standard collection classes (NSArray, NSDictionary, NSSet) and standard value types (NSString, NSDate, NSNumber)
- But there can be custom classes, which typically have names that sound like data-bearing objects

Controller objects

- View and model objects are the factory workers of an application – they focus tightly on specific tasks.
- For example, an instance of UILabel (a view object) knows how to display text in a given font within a given rectangle.
- An NSString instance (a model object) knows how to store a characterstring.
- But the label doesn’t know what text it should display, and the string doesn’t know what characters it should store.

Controller objects Cont….

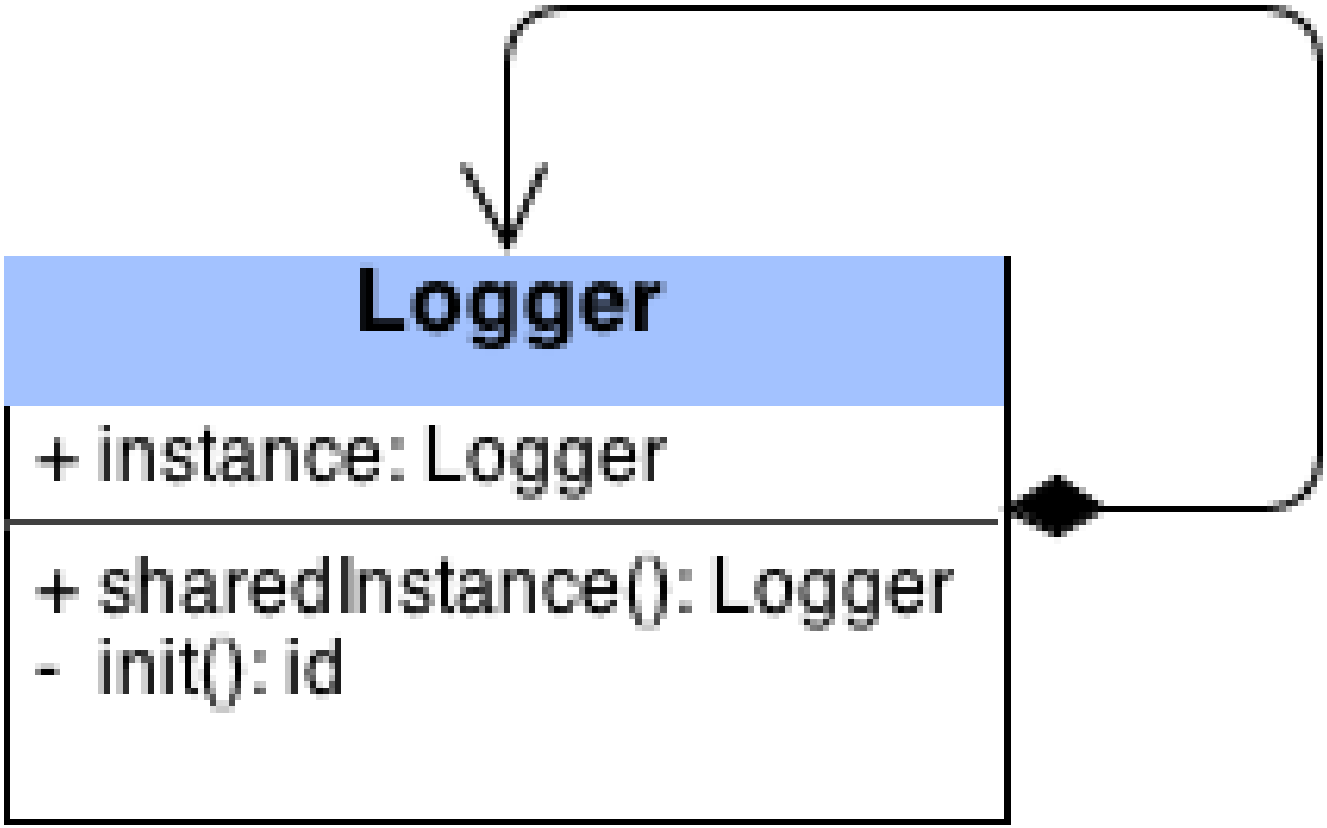
- This is where controller objects come in
- Controllers are the managers in an application
- They keep the view and model objects in sync, control the “flow” of the application
- and save the model objects out to the filesystem


User Input
User interacts with view object
View sends message to controller
Controller updates model objects
Views
Controller
Model
Controller updates view with changes in model objects
Controller takes data from model objects that its views are interested in

The Singleton Pattern

- The Singleton design pattern ensures that only one instance exists for a given class and that there's a global access point to that instance
- It usually uses lazy loading to create the single instance when it's needed the first time
- Apple uses this approach a lot. For example: **[NSUserDefaults standardUserDefaults]**, **[UIApplication sharedApplication]**, **[UIScreen mainScreen]**, **[NSFileManager defaultManager]** all return a Singleton object

How to Use the Singleton Pattern



How to Use the Singleton Pattern…

- The above image shows a Logger class with a single property (which is the single instance), and two methods: **sharedInstance** and **init**
- The first time a client sends the **sharedInstance** message, the property **instance** isn't yet initialized, so you create a new instance of the class and return a reference to it
- The next time you call **sharedInstance**, **instance** is immediately returned without any initialization. This logic promises that only one instance exists at all times

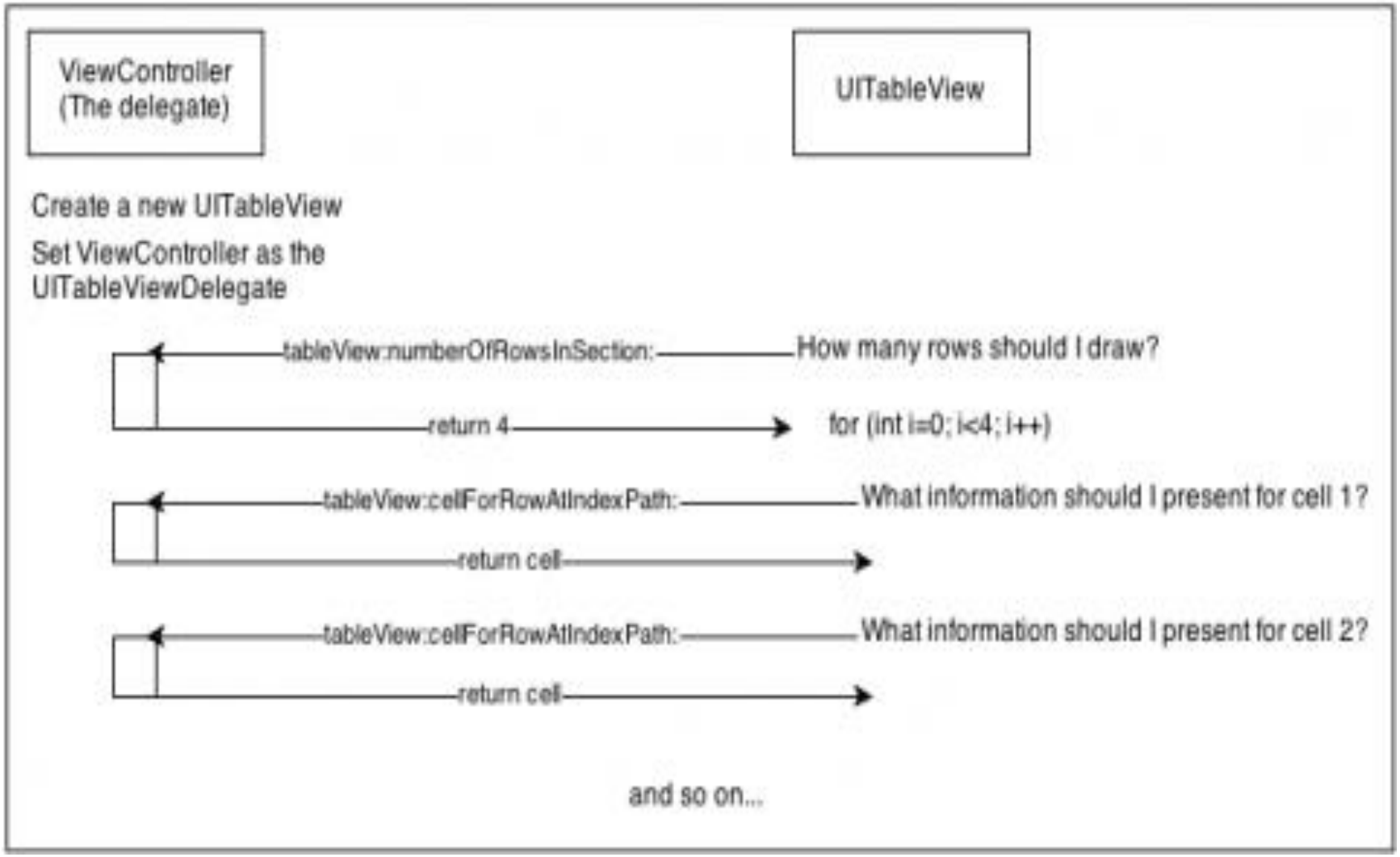
Delegation

- Delegation, is a mechanism in which one object acts on behalf of, or in coordination with, another object
- For example, when you use a **UITableView**, one of the methods you must implement is **tableView:numberOfRowsInSection**
- You can't expect the **UITableView** to know how many rows you want to have in each section, as this is application-specific

Delegation…

- Therefore, the task of calculating the amount of rows in each section is passed on to the **UITableView** delegate
- This allows the **UITableView** class to be independent of the data it displays

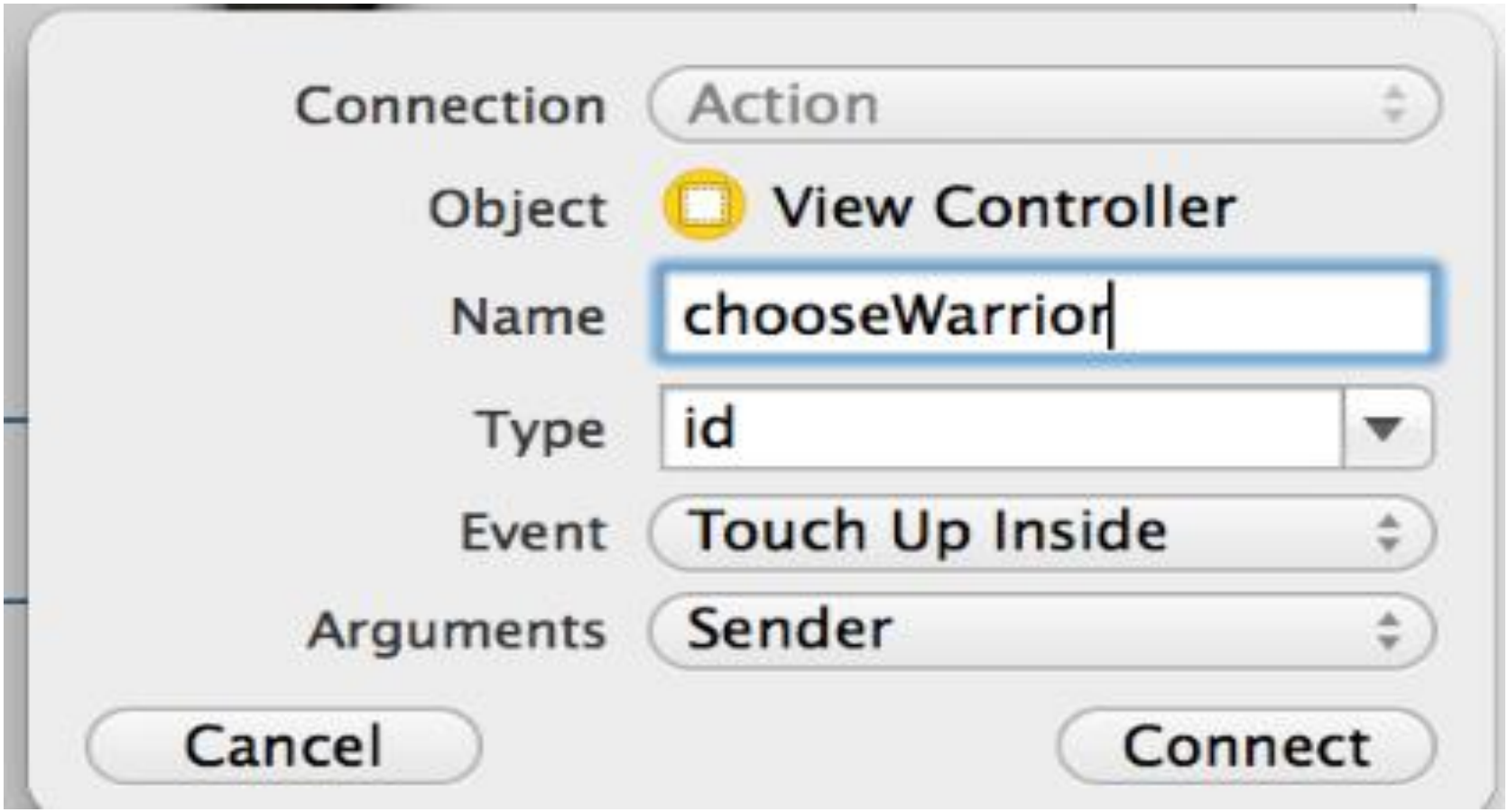
Delegation….


ViewController
(The delegate)
UITableView
Create a new UITableView
Set ViewController as the
UITableViewDelegate
tableView:numberOfRowsInSection:
How many rows should I draw?
return 4
for (int i=0; i<4; i++)
tableView:cellForRowAtIndexPath:
What information should I present for cell 1?
return cell
tableView:cellForRowAtIndexPath:
What information should I present for cell 2?
return cell
and so on...

The Target-Action Mechanism

- The target-action mechanism enables a control object—that is, an object such as a button, slider, or text field—to send a message to another object that can interpret the message and handle it as an application-specific instruction
- The receiving object, or the target, is usually a custom controller object
- The message—named an action message—is determined by a selector, a unique runtime identifier of a method

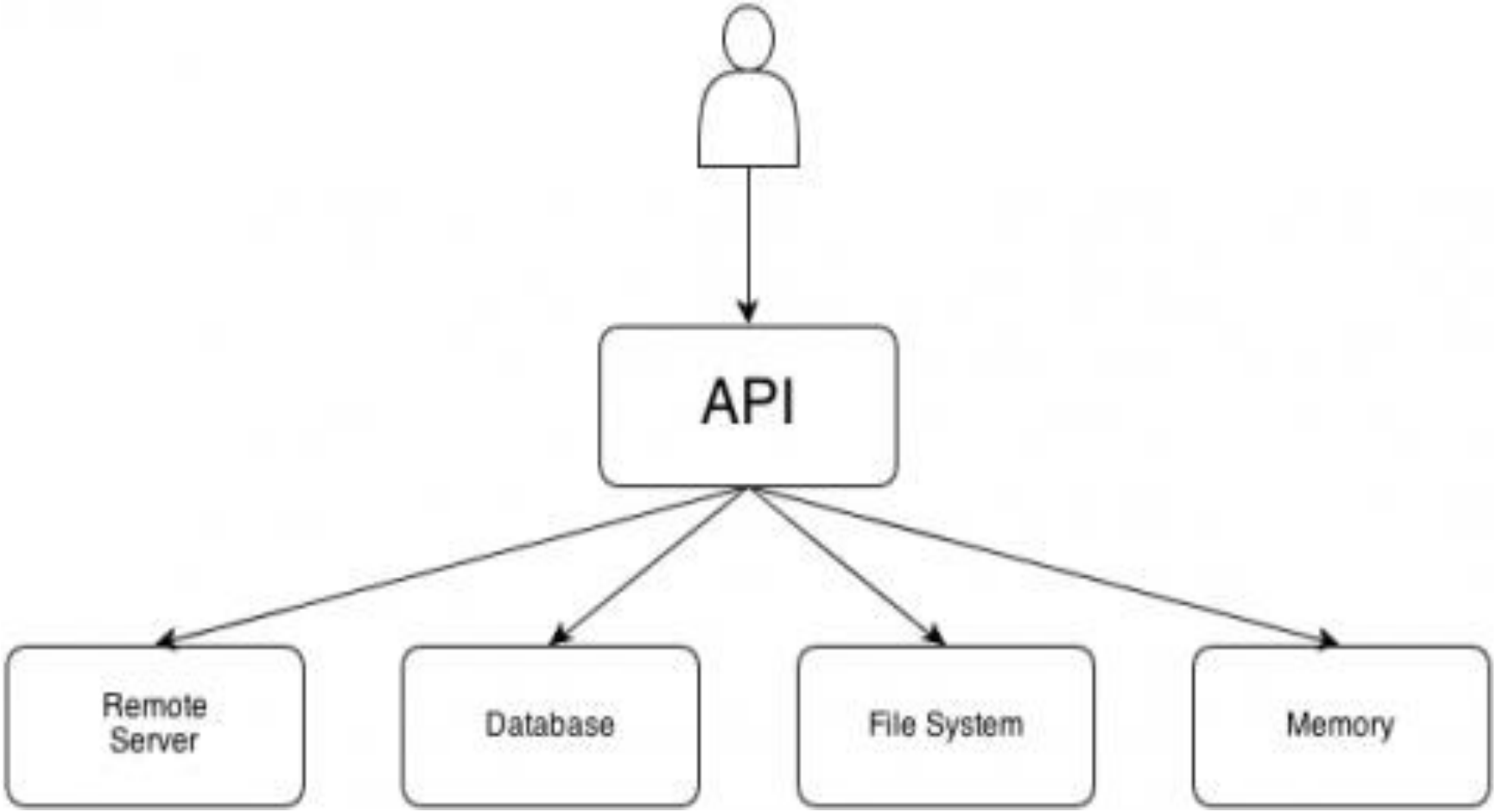
Action on a Button



The Facade Design Pattern

- The Facade design pattern provides a single interface to a complex subsystem
- Instead of exposing the user to a set of classes and their APIs, you only expose one simple unified API
- The user of the API is completely unaware of the complexity that lies beneath
- This pattern is ideal when working with a large number of classes, particularly when they are complicated to use or difficult to understand

The Facade Design Pattern…



Quick Links

- For more on design pattern visit links
- Link 1 https://developer.apple.com/legacy/library/documentation/Cocoa/Conceptual/CocoaFundamentals/CocoaDesignPatterns/CocoaDesignPatterns.html
- Link 2 http://www.raywenderlich.com/46988/ios-design-patterns