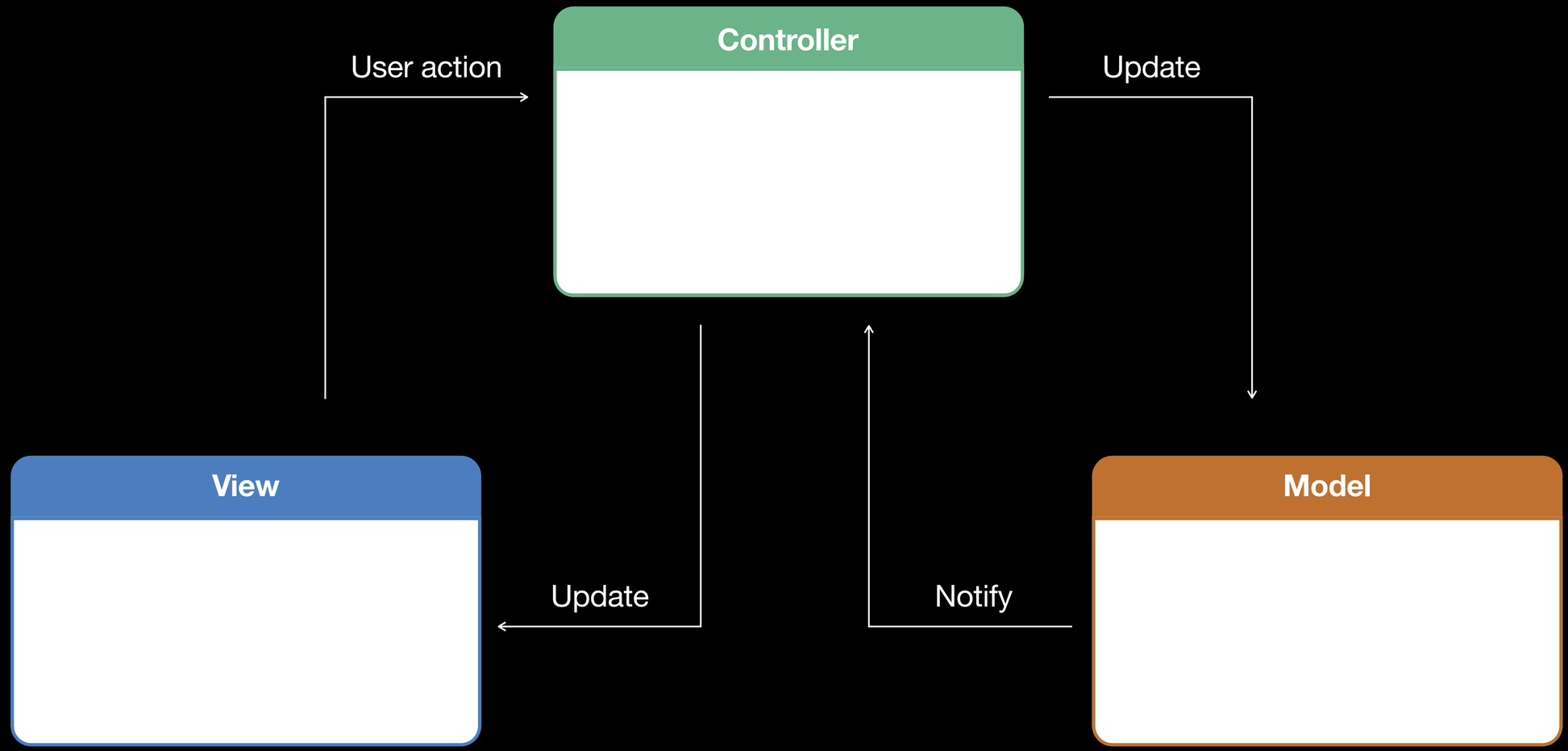
Unit 4—Lesson 3: Model-View-Controller

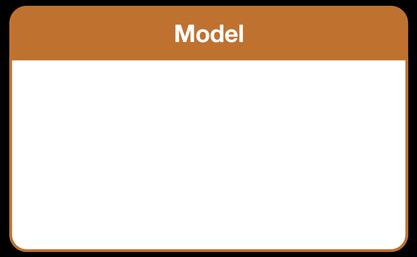
Model-View-Controller
Pattern



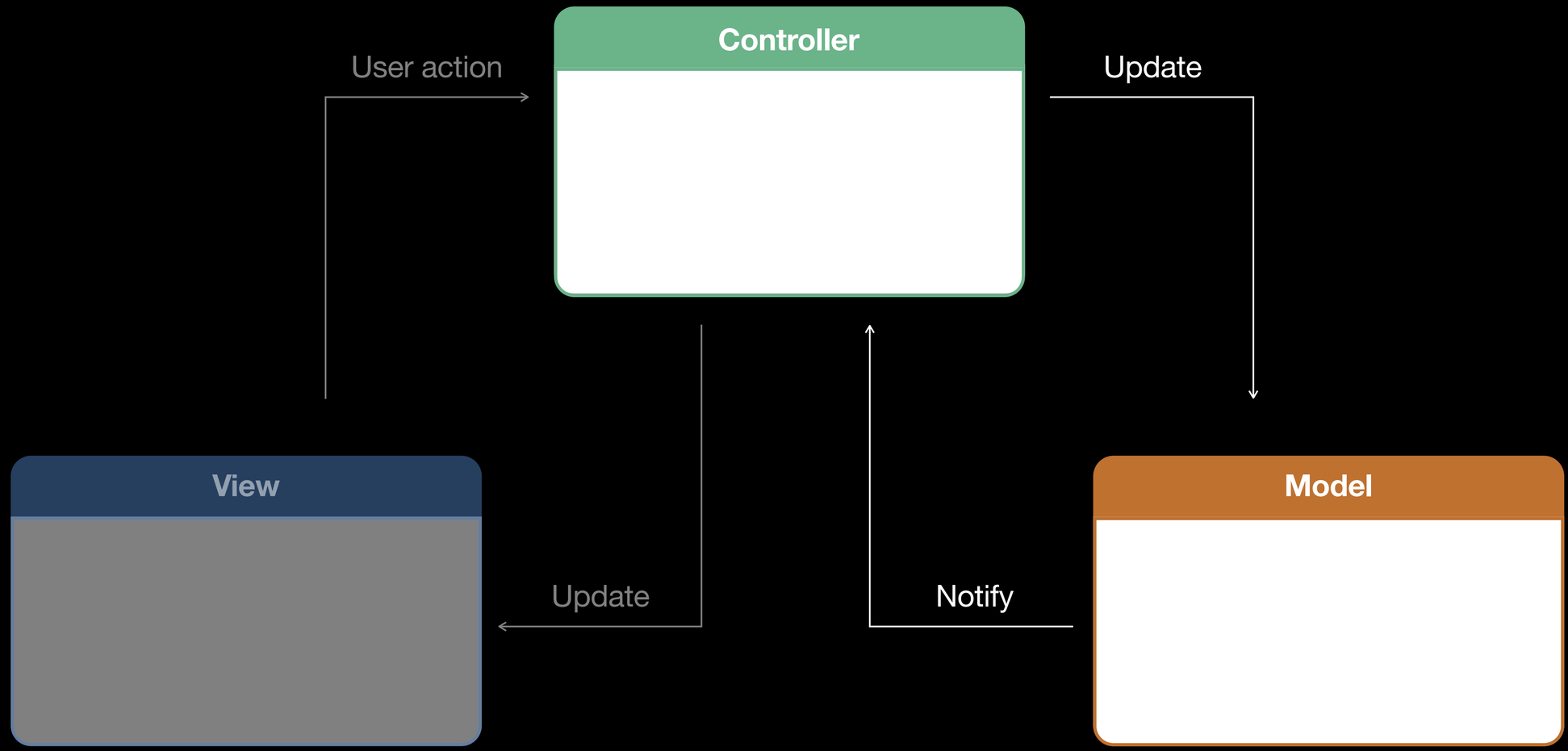
Model objects

Groups the data needed for a specific problem domain or a type of solution to be built

Can be related to other model objects



Model objects
Communication



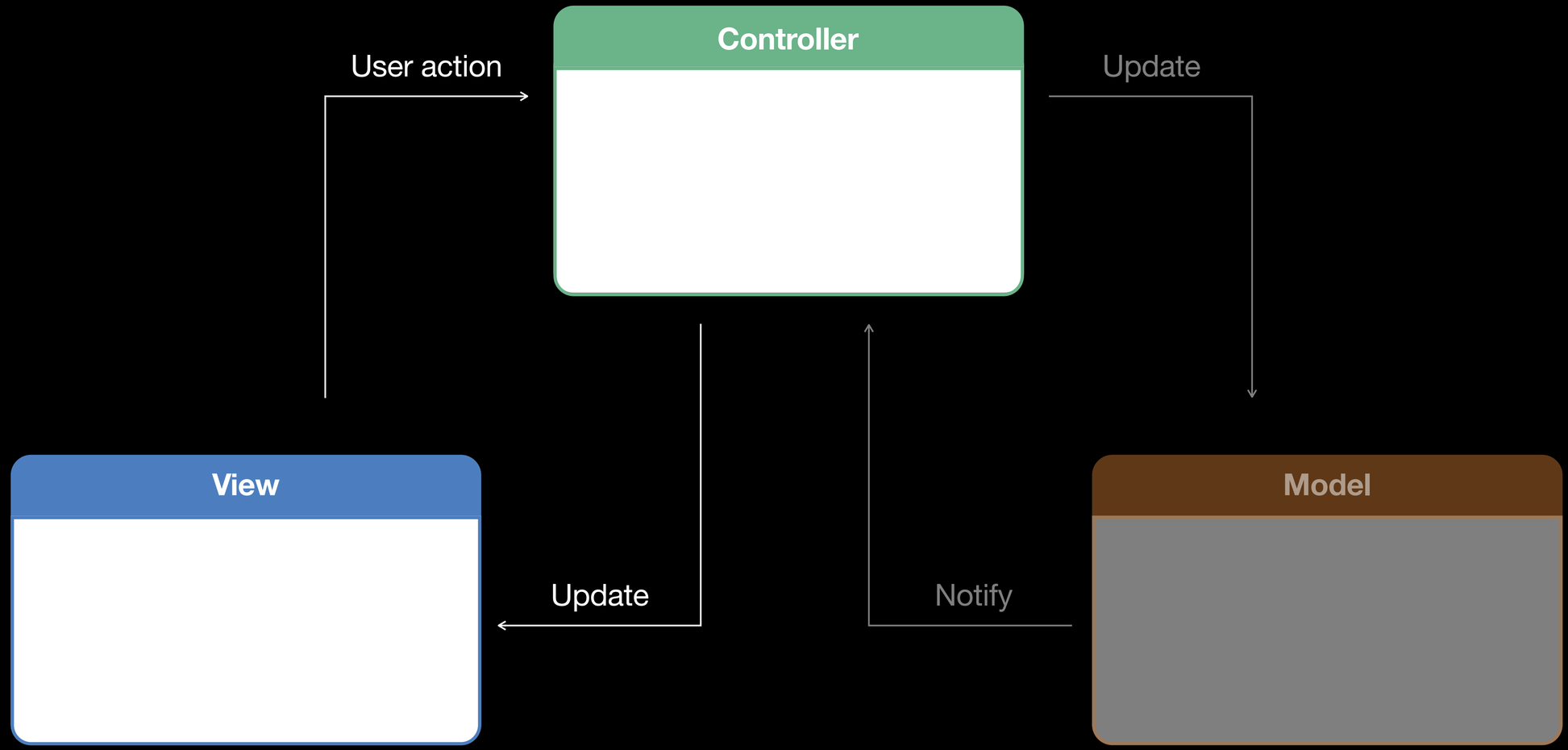
Views

Displays data about the app’s model objects and allows user to edit the data

Can be reused to show different instances of the model data

Views

Communication



Controllers

Acts as the messenger between views and model objects

Types:

- View controllers
- Model controllers
- Helper controllers

Controllers

Model

Helps control a model object or collection of model objects

Three common reasons to create a model controller:

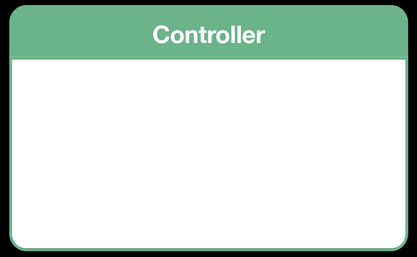
- Multiple objects or scenes need access to the model data
- Logic for adding, modifying, or deleting model data is complex
- Keep the code in view controllers focused on managing the views

Crucial in larger projects for readability and maintainability

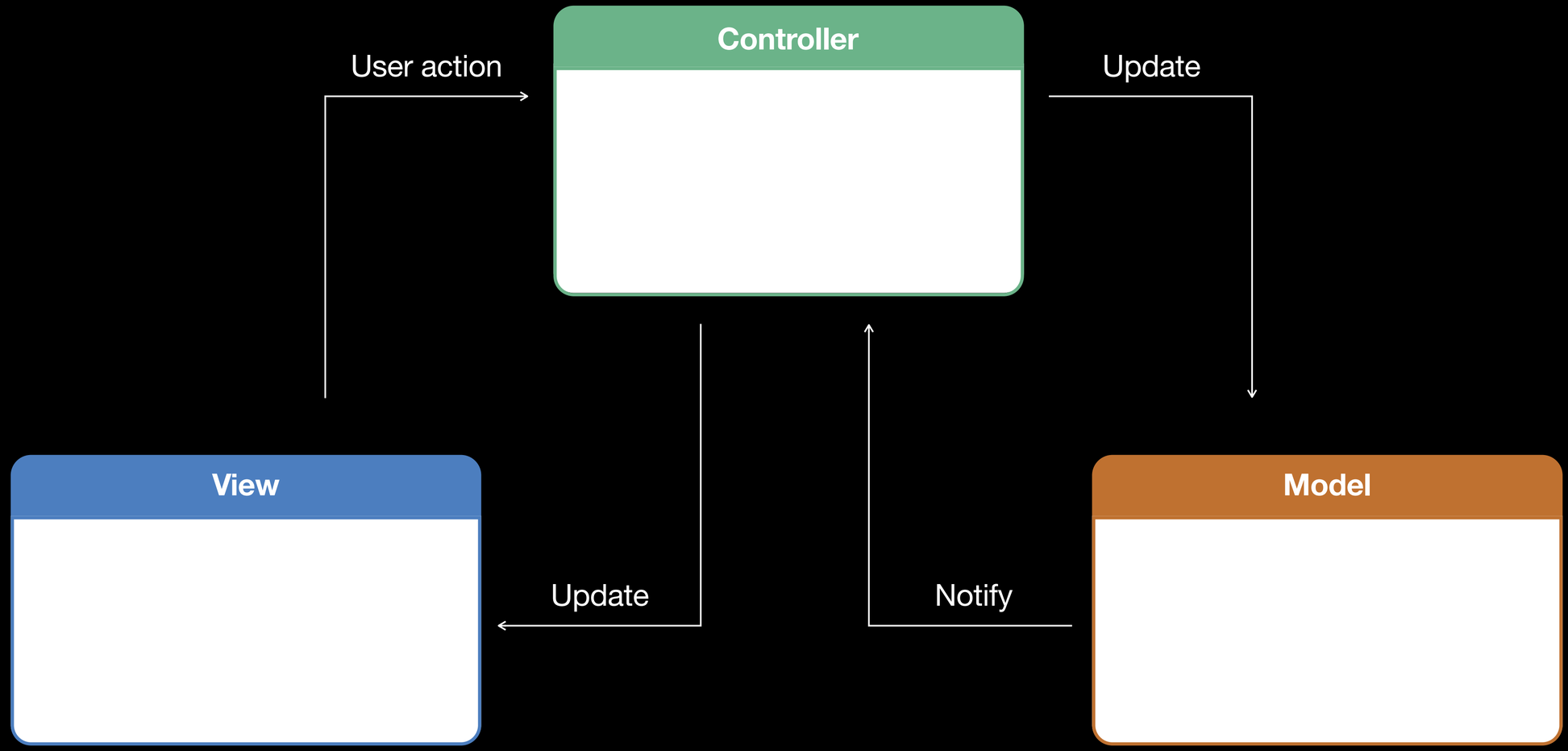
Controllers

Helper

Useful to consolidate related data or functionality so that it can be accessed by other objects in your app



Controllers
Communication



Meal tracker example

Creating an app to track eaten meals

- What should be in a "Meal" model object?
- What views are needed to display meals?
- How many controllers makes sense?

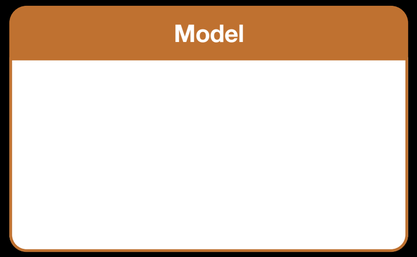
Meal tracker example

Model

Meal:

- Name
- Photo
- Notes
- Rating

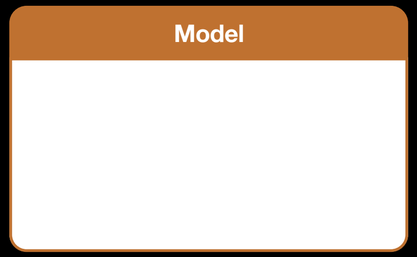
Plus a timestamp



Meal tracker example

Model

```
struct Meal {
    var name: String
    var photo: UIImage
    var notes: String
    var rating: Int
    var timestamp: Date
}
```



Meal tracker example

Views

Two possible views:

- List of all tracked meals
- Details of each meal

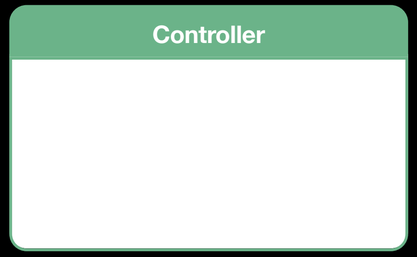
Each needs a view controller class

Meal tracker example

Controllers

Minimum of two controllers:

- List view
- Detail view

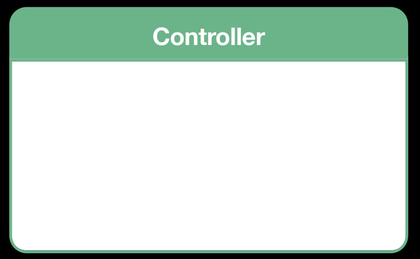


Meal tracker example

Meal list table view controller

```
class MealListTableViewController: UITableViewController {


    var meals: [Meal] = []
    @IBOutlet weak var tableView: UITableView!
}
```



Meal tracker example

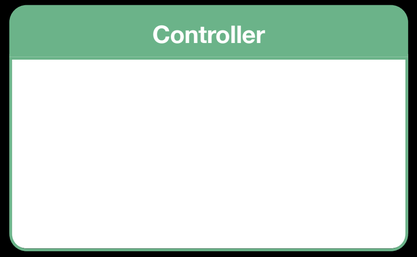
Meal list table view controller

```
class MealListTableViewController: UITableViewController {

    var meals: [Meal] = []

    func saveMeals() {...}

    func loadMeals() {...}
}
```



```swift
class MealListTableViewController: UITableViewController {

    let meals: [Meal] = []

    override func viewDidLoad() {
        // load the meals and set up the table view
    }

    // Required table view methods

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {...}

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {...}
```

```
    // Navigation methods

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Pass the selected meal to the MealDetailViewController
    }
    @IBAction func unwindToMealList(sender: UIStoryboardSegue) {
        // Capture the new or updated meal from the MealDetailViewController and save it to the meals
property
    }

    // Persistence methods
    func saveMeals() {
        // Save the meals model data to the disk
    }

    func loadMeals() {
        // Load meals data from the disk and assign it to the meals property
    }
}
```

Meal tracker example

Meal detail view controller

```
class MealDetailViewController: UIViewController {...}
```

```swift
class MealDetailViewController: UIViewController, UIImagePickerControllerDelegate {

    @IBOutlet weak var nameTextField: UITextField!
    @IBOutlet weak var photoImageView: UIImageView!
    @IBOutlet weak var ratingControl: RatingControl!
    @IBOutlet weak var saveButton: UIBarButtonItem!

    var meal: Meal?

    override func viewDidLoad() {
        if let meal = meal {
            update(meal)
        }
    }

    func update(_ meal: Meal) {
        // Update all outlets to reflect the data about the meal
    }
```

```
    // Navigation methods

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Update the meal property that will be accessed by the MealListTableViewController to update the list of meals
    }

    @IBAction func cancel(_ sender: UIBarButtonItem) {
        // Dismiss the view without saving the meal
    }
```

Reminder

Model-View-Controller is a useful pattern

More than one way to implement it

Everyone has their own style

Yours will evolve as you gain experience

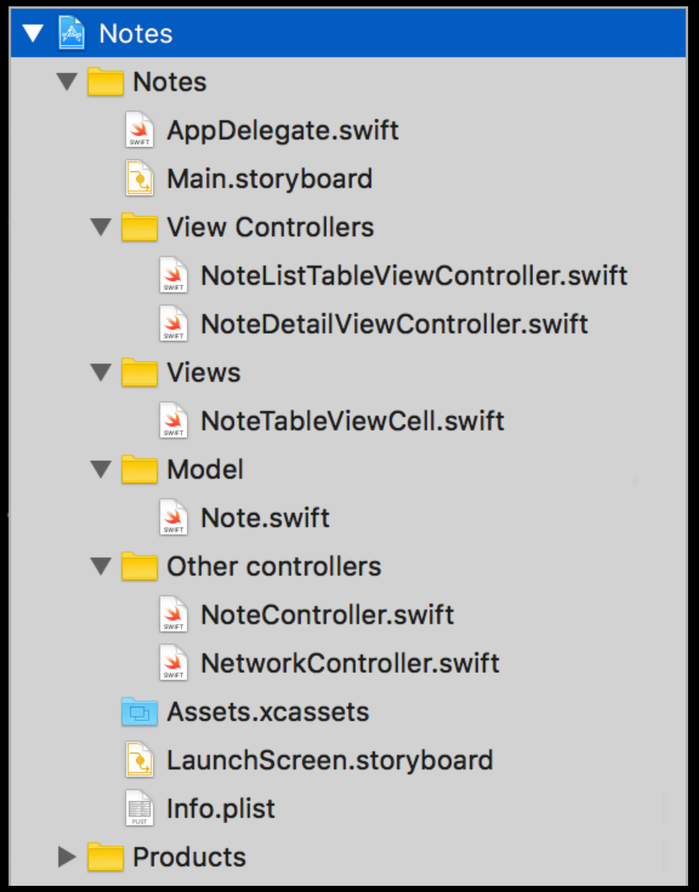
Project organization

Use clear, descriptive filenames

Create separate files for each of your type definitions

Write your code as if complete strangers are going to read it

Group files to help organize your code



Unit 4—Lesson 3

Model-View-Controller

Learn how to organize files, structures, and classes into a design pattern called MVC, which stands for Model-View-Controller.

MVC will help you architect the files in your app, as well as the interactions and relationships between different types and instances.

Unit 4—Lesson 3

Lab: Favorite Athletes

Plan out and create an app that uses proper MVC design. Your app will have two screens for displaying the user's favorite athletes.

It will allow the user to add new athletes to the list, as well as to edit existing entries.