DataSource Protocol 살펴보기

Drake

왜 DataSource Protocol를 알아야 하나?

VendingMachineApp/VendingMachineApp/ManagerViewController.swift

```
268 +     /// 파이정보 프로토콜 준수
269 +     func getPieInfo() -> DrinkPieInfo {
270 +         return vendingMachine.getPieInfo()
271 +     }
```

godrm 25 days ago Contributor

프로토콜을 나눌때는 이렇게 한번에 가져가는 게 아니라
예를 들어 DrinkNameInfo를 여러번에 나눠서 가져가도록 함수를 쪼개는 게 좋습니다.
그런 방식을 뷰의 데이터를 구하는 프로토콜이라고 해서 DataSource 프로토콜 방식이라고 합니다.
곧 UITableView 나 UICollectionView 를 만들때 그대로 활용하게 됩니다.
마무리단계에서 개선해보세요.

- JK 가 쓰라고 함
- 데이터를 넘겨줄때 통째로 주는 방법은 좋지 않음

이전 과제 진행 과정

- 원래 방식 : 음료정보를 가지고 있는 객체를 통으로 넘겨줌
- JK 제안 : DataSource Protocol 사용
- 수정된 방식 : Protocol 을 이용해서 정보객체 넘겨줌
- 마음이 급해서 DataSource 는 안씀
- 발표를 기회로 조사해서 다음 과제에 적용할 예정

DataSource Protocol 이란?

- 사용되는곳 :

  UICollectionViewDataSource,

  UIPickerViewDataSource 등등

- 사용되는 방식 : Delegate 와 함께

- Delegate?

Delegate?

- 사전적 정의 : 대리자, 위임자
- 검색하면 나오는 내용 : 앱 생명주기, 데이터 전달 방식
- iOS 에서 객체간 데이터 전송시 쓰이는 방법
- 객체가 직접 일을 하지 않고 델리게이트를 통해서 일을 시키는 방식
- 프로토콜을 채택해서 진행하는 방식과 유사

Delegate 예시

Delegate 예시

- 델리게이트 선언

```
protocol FirstVCDelegate {
  func passData(data: String)
}
```

- 델리게이트를 소유하는 객체 선언

```
class FirstVC {
 var delegate: FirstVCDelegate?
}
```

Delegate 예시

- 델리게이트 역할의 객체 선언

```
class SecondVC: FirstVCDelegate {
 func passData(data: String) {
  print("Something happened")
 }
}
```

Delegate 예시

- 객체 생성

```
let firstVC = FirstVC()
let secondVC = SecondVC()
```

- 델리게이트 부여

```
firstVC.delegate = secondVC  // secondVC = delegate
```

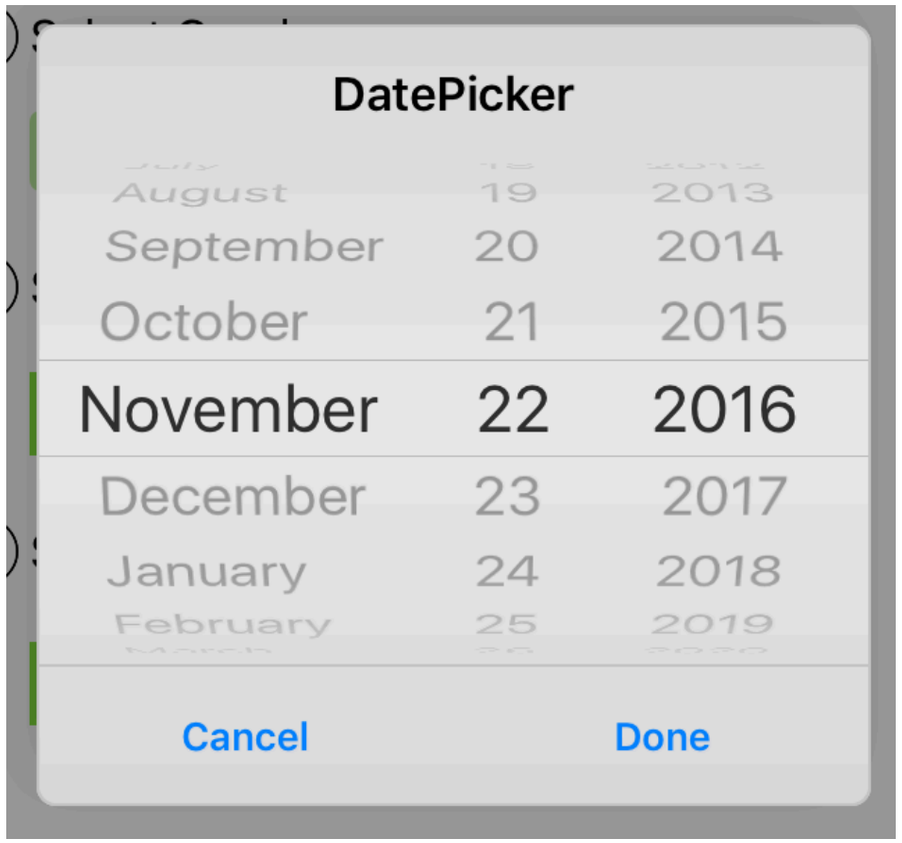
- 델리게이트를 통한 작업 진행

```
firstVC.delegate?.passData(data: "a bunch of contracts”)
// "Something happened"
```

DataSource 예시

DataSource 예시

- PickerView



UIPickerView 정의

Providing the Picker Data

`var dataSource: UIPickerViewDataSource?`

The data source for the picker view.

`protocol UIPickerViewDataSource`

The `UIPickerViewDataSource` protocol must be adopted by an object that mediates between a `UIPickerView` object and your application's data model for that picker view. The data source provides the picker view with the number of components, and the number of rows in each component, for displaying the picker view data. Both methods in this protocol are required.

Customizing the Picker Behavior

`var delegate: UIPickerViewDelegate?`

The delegate for the picker view.

`protocol UIPickerViewDelegate`

The delegate of a `UIPickerView` object must adopt this protocol and implement at least some of its methods to provide the picker view with the data it needs to construct itself.

UIPickerView protocol

- dataSource

```
var dataSource: UIPickerViewDataSource?
```

The data source for the picker view.

- delegate

```
var delegate: UIPickerViewDelegate?
```

The delegate for the picker view.

- 이미 존재. 새로 만들 필요 없음

DataSource 정의

- 정의된 함수들

```
public protocol UIPickerViewDataSource : NSObjectProtocol {


    // returns the number of 'columns' to display.
    @available(iOS 2.0, *)
    public func numberOfComponents(in pickerView: UIPickerView) -> Int


    // returns the # of rows in each component..
    @available(iOS 2.0, *)
    public func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int)
        -> Int
}
```

- 프로토콜을 채택한 객체가 해당 함수를 선언해서 사용

- pickerView 개수,한 칸에 몇개인지

DataSource 선언

```
public func numberOfComponents(in pickerView: UIPickerView) -> Int {
    return 1
}

public func pickerView(_ pickerView: UIPickerView,
numberOfRowsInComponent component: Int) -> Int {
    return Array.count

}
```

pickerView 정의 및 선언

- 입력받은 행의 내용 출력

```
@available(iOS 2.0, *)
optional public func pickerView(_ pickerView: UIPickerView, titleForRow row: Int,
    forComponent component: Int) -> String?
```

- 선언한 배열을 불러오도록 수정

```
func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent
component: Int) -> String? {
        return Array[row]
    }
```

정리

- DataSource 는 Delegate 를 이용하는 방법중 하나
- Delegate 는 객체가 일을 다른 객체에게 맡기는 방식
- 이미 정의된 기능에서 delegate 와 함께 사용
- DataSource 는 데이터의 정보를 담당. ex> Count, length ..
- delegate 에서 DataSource 의 정보를 받아서 출력

다음 코드파티 주제 후보

- DataSource Protocol 을 실제로 적용
- 클로저란 무엇인가
- UIView 이동 및 위치지정

사족

- 코드파티의 장점 :

    1. 몰랐는데 시간이 없어서 공부하지 못한 부분 정리 가능
    2. 블로그에 쓸 내용 생김
    3. 취업에 유리

출처

- BTD : https://www.bobthedeveloper.io/blog/the-complete-understanding-of-swift-delegate-and-data-source
- Zedd : https://zeddios.tistory.com/10?category=682195