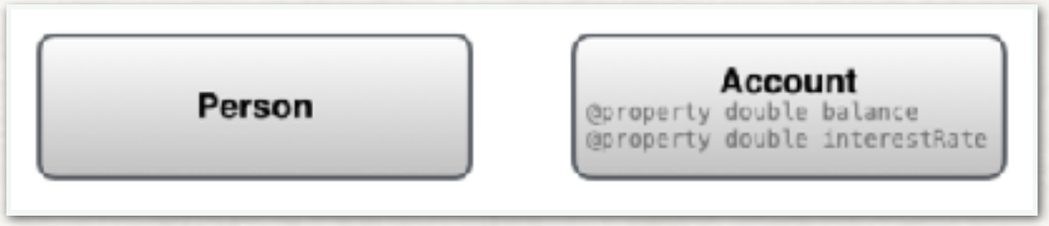
WEEK 2

KVO / DELEGATE

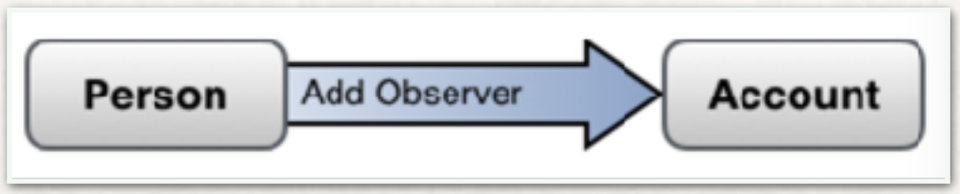
윤사라

KVO(Key Value Observing)

- 한 객체가 다른 객체 프로퍼티의 변경된 값을 확인할 수 있도록 함
- 어플리케이션에서 모델과 컨트롤러 레이어들간의 커뮤니케이션에 유용하다.
- 일번적으로 컨트롤러 객체가 모델 객체의 프로퍼티를 observe하며 뷰 객체는 컨트롤러를 통해 모델 겍체 속성을 observe 한다.
- 만약 Person이라는 객체가 은행 Account객체와 연결되어 있다고 가정할 때, Account객체에서 변경되는 값을 일일이 Person이 주기적으로 확인해야 하면 실용적이지 못하다. 이런 상황에서 KVO를 통해 값이 변경될 때마다 알림을 받는 것이 효율적일 것이다.



- KVO를 사용하려면 먼저 observed 객체인 account의 속성을 생성한 후, observer 인스턴스 Person과 observed instance인 account를 등록한다. Person은 addObserver:forKeyPath:options:context: 메시지를 Accout에 보낸다.



- 만일 Account로 부터 값 변동에 대한 알림을 받기 위해서는 Person은 observeValueForKeyPath:ofObject:change:context: 메소드를 구현하고, 등록된 key path가 변경될때마다 Person에게 메시지가 오도록 한다. 변경 알림에 따라 Person객체는 적합한 액션을 취한다.
- 변경알림을 더이상 받지 않기 위해서는 Person 인스턴스는 Account 객체에게 removeObserver:forKeyPath: 메시지를 통해 삭제해야 한다.

KVO vs Delegate

KVO(Key Value Observing)

***장점**

- 두 객체간의 정보를 sync하기 편리하다.
- observing하는 값의 새로운 값과 이전 값의 프로퍼티를 제공함

***단점**

- observe를 원하는 property에서 Strings로 정의되므로 compile time warning/checking이 없다
- 할당을 취소할 때, observer를 제거해야 함

Delegate

***장점**

- 모든 이벤트들이 delegate 프로토콜에 명백히 정의되어 있는 strict syntax
- Control Flow를 알기 쉽기 때문에 tracking이 수월하다.
- 하나의 컨트롤러에 여러 프로토콜을 정의할 수 있다.
- delegate에 의한 메소드가 구현되지 않았을 때, Complie time warnings/errors

***단점**

- 정의해야 할 많은 코드가 필요하다. 1) 프로토콜 정의 2)컨트롤러에 delegate프로퍼티 정의
  3) delegate 메소드 정의 implementation
- 객체 할당 취소 시, delegate를 nil로 설정할 때 주의
  할당이 취소된 객체를 메소드를 통해 호출하면 메모리 크래쉬 발생