BoostCamp 2 주차 발표

Delegation Pattern

Delegation 패턴의 최고장점
=> 엄격한 구문 !

Delegation Pattern

확실하게 정의된 델리게이트 프로토콜의 함수들만을 사용하여 이벤트를 처리

Delegation Pattern

프로토콜을 준수하지 않으면 ?
=> 컴파일 타임에 에러를 낼 수 있다 !!

Delegation Pattern

해당 컨트롤러의 범위 내에서만 프로토콜이 정의됨 => 어플리케이션의 통제흐름을 파악하기가 매우 용이함

Delegation Pattern

프로토콜 메소드로부터 리턴값을 받아낼 수 있음 => 델리게이트가 뷰컨트롤러에 도움이되는 정보를 돌려줌

Delegation Pattern

그러나 ...

Delegation Pattern

기능 좀 몇개 쓰려는데 ...

Delegation Pattern

정의해야될 것이 많다

1. 프로토콜 정의
2. 델리게이트를 위한 프로퍼티 (뷰 컨트롤러 내에)
3. 델리게이트 메소드 안의 구현코드 작성

Delegation Pattern

조심해야될 것이 많다

객체 할당 해제시 델리게이트 nil 설정

=> 실패하면 해제된 객체에서의 메소드 호출로 메모리 충돌발생 가능

KVO

Key Value Observing
=> 특정 객체의 프로퍼티의 값을 관찰하여 변화가 일어났을 경우를 감지

KVO

특정 객체의 변수값에 이벤트를 등록한다 ??

KVO

프로퍼티의 값의 변화를 감지함으로써 여러 객체간의 동기화를 간편하게 구성할 수 있음 .

KVO

직접 작성하거나 생성하지 않은 객체나 접근 권한이 없는 객체의 변화에 대처가 가능

KVO

관찰중인 프로퍼티의 변화 전과 후의 값을 모두 살펴볼 수 있어 변화된 상황에 맞게 반응을 취할 수 있음

KVO

하지만 ..

KVO

관찰하고자 하는 프로퍼티들은 문자열을 이용해 정의됨 => 컴파일타임 경고를 확인해볼 수 없음 => 에러 확인이 용이하지 않음

KVO

다수의 프로퍼티의 상태변화를 관찰할 때 복잡한 if 구문으로 처리해야함

KVO

결론 : KVO 의 남발은 어플리케이션의 흐름을 파악하는 것이 용이하지 않게 함

감 사 합 니 다