

Let's us go 2018 - Summer

iOS TDD 실무에 적용하기

유금상 (AnyObject)

iOS TDD 실무에 적용하기

유금상 (AnyObject)

iOS TDD 실무에 적용하기

유금상 (AnyObject)

!! 개인의 경험과 느낌에 근거해
작성되었기 때문에 실제로 작동
하지 않을 수 있습니다.

팀 설득하기

미지에 대한 두려움

“모르는”

것에 대한 두려움

“익숙하지 않은”
것에 대한 두려움

“안 해본”

것에 대한 두려움

“새로운”

것에 대한 두려움

거부감

겨부감

지식

지식

지식을 전달하는 방법

팀 설득하기

동료 설득하기

다양한 타입의 개발자

다양한 타입의 개발자

다양한 타입의 개발자

TDD를 실패한 경험

다양한 타입의 개발자

TDD를 알기만

다양한 타입의 개발자

새로운 것에 무관심

지식을 전달하는 방법

지식을 전달하는 방법
흔히 하는 실수

흔히 하는 실수

흔히 하는 실수

왜 내 말을 들어주지 않을까

흔히 하는 실수

우리팀은 기술적 진보에
관심이 없어

흔히 하는 실수

동료들은 실력이 부족해

흔히 하는 실수

우리팀은 최악이야

흔히 하는 실수

어떤 노력을 했는지 생각
해볼 필요가 있다

흔히 하는 실수

설득에 필요한
충분한 개발적 역량을
가지고 있는가?

흔히 하는 실수

그 역량이 동료들에게
신뢰를 주고 있는가?

다양한 타입의 개발자 설득하기

가르치는 느낌이 들면 안됨

존중 받는 느낌이 들게

다양한 타입의 개발자 설득하기

함께 찾아내보자

다양한 타입의 개발자 설득하기

철저한 준비가 필수

다양한 타입의 개발자 설득하기

함께 베스트 프랙티스를
찾아보자고 제안

다양한 타입의 개발자 설득하기

페어코딩 / 몹코딩

다양한 타입의 개발자 설득하기

단, 구성원 모두
실무에 적용할 만한
지식과 경험이 필요함

다양한 타입의 개발자 설득하기

신뢰하는 관계

팀 설득하기

상사 설득하기

상사보다
동료를
먼저 설득해야 함

상사보다
동료를

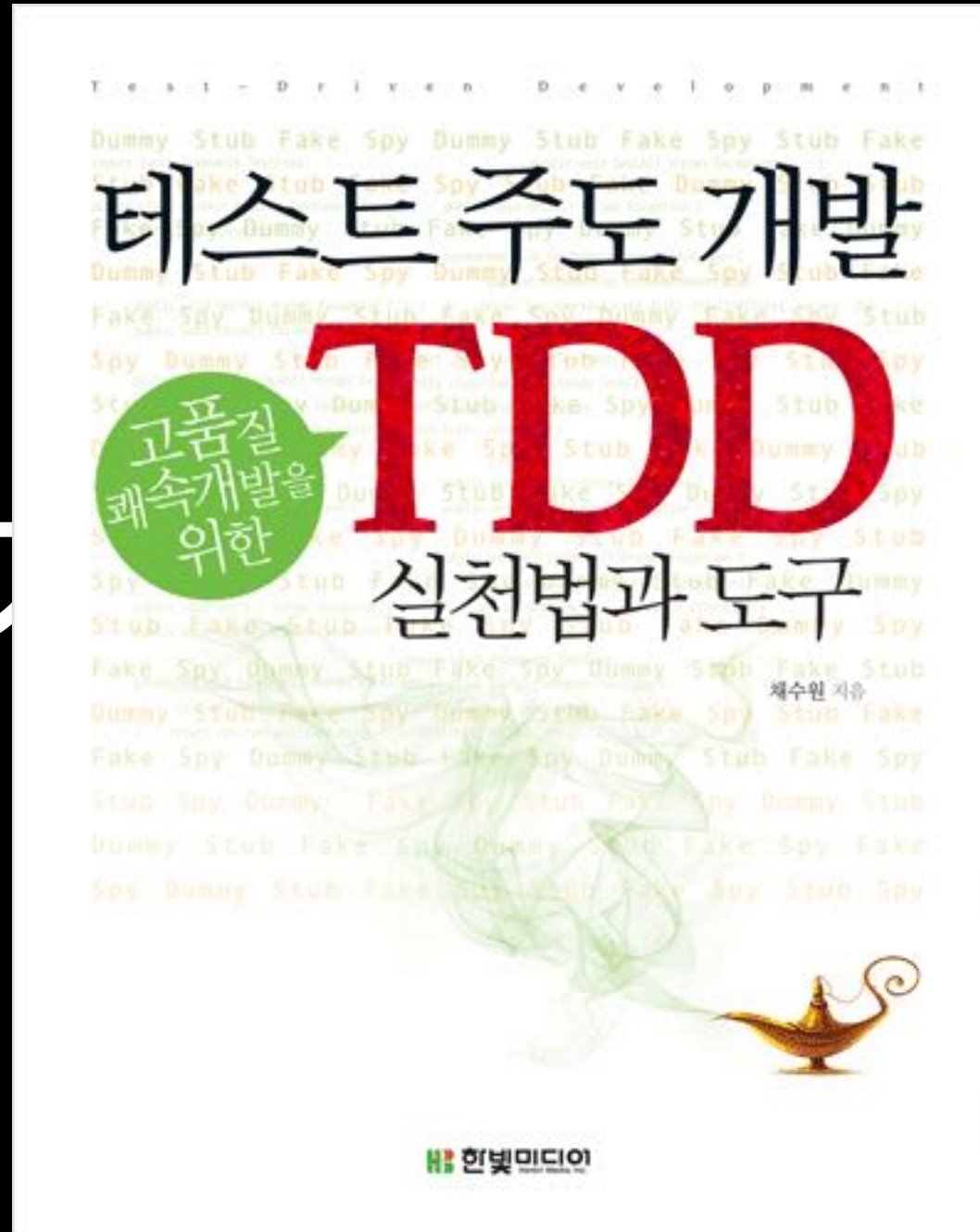
먼저 설득해야 함

동료들이 마음으로 받아들이지
못하면 강제로 시켜도
절대 성공하지 못한다.

상사 설득 하기

기술적 완성도

상사 설득 하기



상사 설득 하기

1. 더 정확한 일정

상사 설득 하기

팀장님 : 이거 얼마나 걸릴 거 같아요?

개발자 : 대충.... 2주면 될 거 같습니다.

상사 설득 하기

2. 내가 없어도 돌아간다.

상사 설득 하기

3. 대외적인 팀의 실력 어필

상사 설득 하기

4. 기술적 완성도

상사 설득 하기

단, 상사가
실무에 적용할 만한
지식과 경험이 필요함

상사 설득 하기

신뢰하는 관계

안드로이드 개발자 설득하기

안해도 됨

팀 설득하기

다른 직군 설득하기

다른 직군 설득 하기

버그가 반복되지 않을 것

다른 직군 설득 하기 버그가 반복되지 않을 것

다른 직군 설득 하기 버그가 반복되지 않을 것

조건 한정을 잘해야 함

다른 직군 설득 하기 버그가 반복되지 않을 것

버그가 전혀 없을 것이다.

다른 직군 설득 하기 버그가 반복되지 않을 것

~~버그가 전혀 없을 것이다.~~



다른 직군 설득 하기 버그가 반복되지 않을 것

같은 버그가
같은 원인으로
반복되지 않을 것이다.



다른 직군 설득 하기 버그가 반복되지 않을 것

다른 곳을 고쳤을 때,
이전 버그가
재발생하지 않을 것이다.



다른 직군 설득 하기

신뢰하는 관계

공통점

신뢰

카카오의 일하는 방식

카카오의 일하는 방식

신충현

카카오의 일하는 방식

신뢰 충돌 헌신

카카오의 일하는 방식

신뢰

신뢰

동기에 대한 신뢰
역량에 대한 신뢰

신뢰는 어디에서 오는가?

공짜로 생기는 것은 아님

우리 회사에 들어왔으니까 ?

오랜 기간 서로 노력해야만
얻을 수 있다.

일관성 있게
솔직한 태도

윗분들은 조직 개편을 통해
효율을 올리고 싶어하지만..
글썩요..

신뢰 유지 하기

신뢰 유지 하기

프로젝트 중

신뢰 계속 유지하기

꼼꼼함 없는 커뮤니케이션

중간 공유

중간 공유



신뢰 유지 하기

평소에 신뢰를 쌓는 방법

신뢰 유지 하기 > 평소에

일관성

신뢰 유지 하기 > 평소에

인간관계

뺑치지 말것

뻥치지 말것

모르는 것 인정하기

무조건 안된다고 하지 않기

다른 직군에게도
가능하면 자세하게 설명하기

TMI??

**자세히 알 수록
공포는 줄어든다.**

실무에 적용

계획하기

from XP

eXtreme Programming

계획하기

가능하면 자세하게

계획하기

배먹지 말 것

계획하기

유지하기 /

지속적인 계획 업데이트 하기

생각치 못한 부분

변경되는 기획

예상치 못한 디자인

내 맘 같지 않은 API

디자인, 기획, 서버가 완전히 끝
나야 개발을 시작할 수 있다고
생각해서는 안된다.

iOS TDD 실무에 적용하기

유금상 (AnyObject)

환경 셋팅 하기

유닛 테스트를 돌리는
기본적인 방법

Choose options for your new project:

Product Name:

Team:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

- Use Core Data
- Include Unit Tests
- Include UI Tests

Cancel

Previous

Next

Choose options for your new project:

Product Name:

Team:

Organization Name:

Organization Identifier:

Bundle Identifier: com.yourcompany.ProductName

Language:

Use Core Data


Include Unit Tests

Include UI Tests

Cancel

Previous

Next

 + U

주기가 빨라야 한다.

덩치가 큰 프로젝트는
유닛 테스트 실행이 느리다.

시뮬레이터에 앱이
실행/종료되는
과정이 포함됨


테스트 타겟을
메인 타겟에서 분리해서 실행

PROJECT

 Example

TARGETS

 Example

 ExampleTests

 ExampleUITests

▼ **Testing**

Host Application

 Example

Allow testing Host Application APIs

▼ **Signing**

Automatically manage signing

Xcode will create and update profiles, app IDs, and certificates.

PROJECT

Example

TARGETS

Example

ExampleTests

ExampleUITests

▼ **Testing**

Host Application: None

Allow testing Host Application APIs

▼ **Signing**

Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

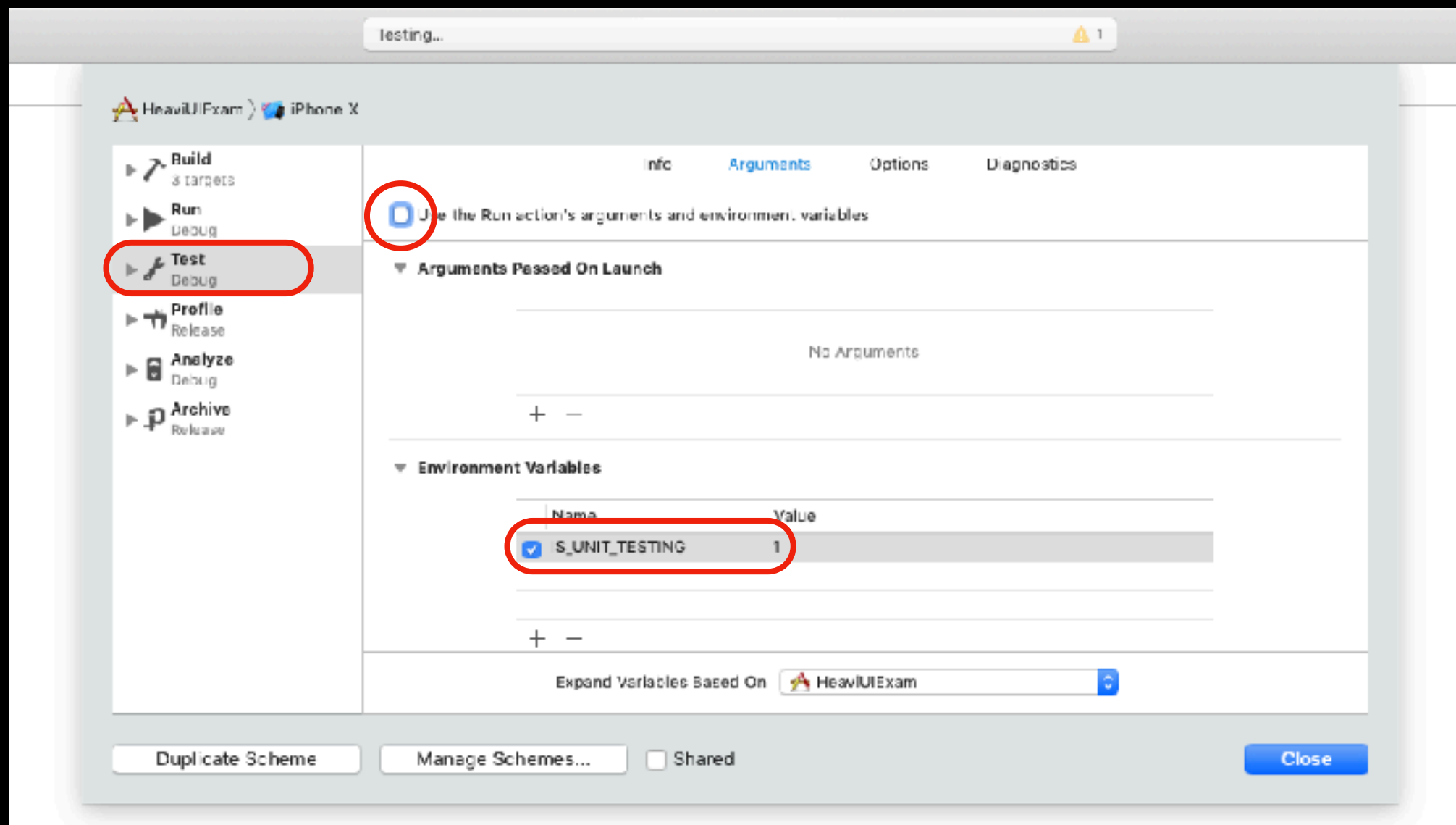
모하비 베타 30에서 안됨

모하비 베타 30에서 안됨

40에서도 안됨 πππ

WWDC 2018

417 Testing Tips & Tricks



```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
    // Override point for customization after application launch.
    let isUnitTesting = ProcessInfo.processInfo.environment["IS_UNIT_TESTING"] == "1"
    guard isUnitTesting == false else { return true }

    heavyAppPrepareTask()
    heavyUIJob()

    return true
}
```

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
    // Override point for customization after application launch.
    let isUnitTesting = ProcessInfo.processInfo.environment["IS_UNIT_TESTING"] == "1"
    guard isUnitTesting == false else { return true }

    heavyAppPrepareTask()
    heavyUIJob()

    return true
}
```

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
    // Override point for customization after application launch.
    let isUnitTesting = ProcessInfo.processInfo.environment["IS_UNIT_TESTING"] == "1"
    guard isUnitTesting == false else { return true }

    heavyAppPrepareTask()
    heavyUIJob()

    return true
}
```

딱 요 시간 만큼 줄어듦

말 나온 김에..

HeaviUIExam > iPhone X

- ▶ **Build**
3 targets
- ▶ **Run**
Debug
- ▶ **Test**
Debug
- ▶ **Profile**
Release
- ▶ **Analyze**
Debug
- ▶ **Archive**
Release

Info Arguments Options Diagnostics

Build Configuration: **Debug**

Debugger: Debug executable

Debug Process As: Me (vbmania-mbpr)
 root

Tests	Enabled
▶ HeaviUIExamTests	Options... <input checked="" type="checkbox"/>
▶ HeaviUIExamUITests	

+ -

- Execute in parallel on Simulator
- Randomize execution order
- Automatically include new tests

Location: **None**

Application Data: **None**

Close

Duplicate Scheme

Manage Schemes...

Shared

결과



상황별 TDD 방법

레거시 코드

레거시 코드에
덧붙이는 코드

이미지 피커
이미지 업로드

이미지 피커 클래스 - objc

```
- (void)showAlbumWith:(UIViewController *)viewController  
    complete:(ImagePickerCompleteBlock)complete;
```

```
class RxImagePickerTests: XCTestCase {
    func testCanPickImage() {
        let vc = UIViewController()
        let picker = RxImagePicker(owner: vc)
        let observable = picker.pickImage()
    }
}
```

어떤 식으로 쓰고 싶은지를 먼저 테스트 코드로 만들어 봅니다.

```
func testCanPickImage() {
    let vc = UIViewController()
    let picker = RxImagePicker(owner: vc)
    let observable = picker.pickImage()

    var resultImage: UIImage?
    observable.subscribe(onSuccess: { (image) in
        resultImage = image
    }) { (_) in
        XCTFail()
    }

    expect(resultImage).toEventuallyNot(beNil())
}
```

아직 아무것도 없지만 이런식으로 쓸 것 같습니다.


```
class RxImagePicker {
    init(owner: UIViewController) {

    }

    func pickImage() -> Single<UIImage> {
        return Single.just(UIImage())
    }
}
```

클래스를 일단 이렇게 만들면 에러가 사라집니다.

```
class RxImagePicker {
    init(owner: UIViewController) {
    }

    func pickImage() -> Single<UIImage> {
        return Single.just(UIImage())
    }
}
```

```
class RxImagePickerTests: XCTestCase {

    func testCanPickImage() {
        let vc = UIViewController()
        let picker = RxImagePicker(owner: vc)
        let observable = picker.pickImage()

        var resultImage: UIImage?
        observable.subscribe(onSuccess: { (image) in
            resultImage = image
        }) { (_) in
            XCTFail()
        }

        expect(resultImage).toEventuallyNot(beNil())
    }
}
```

```
func testCompareImageInstance() {
    let expectedImage = UIImage()
    let notExpectedImage = UIImage()

    let underTest = RxImagePicker(owner: UIViewController())
    var resultImage: UIImage?

    underTest.pickImage().subscribe(onSuccess: { (image) in
        resultImage = image
    }) { (_) in
        XCTFail()
    }.disposed(by: disposeBag)

    expect(resultImage).toEventually(equal(expectedImage))
    expect(resultImage).toEventuallyNot(equal(notExpectedImage))
}
```

테스트를 하나 더 추가 합니다.
이번엔 이미지 객체로 비교할 수 있는지..
맞는 이미지가 오는 걸 검증할 수 있는지 궁금합니다.

```
let expectedImage = UIImage()
class RxImagePicker {
    init(owner: UIViewController) {
    }

    func pickImage() -> Single<UIImage> {
        return Single.just(expectedImage)
    }
}
```

이미지 객체를 글로벌 영역으로 옮기고
RxImagePicker가 해당 이미지를 반환하도록 합니다.

```

let expectedImage = UIImage()
let notExpectedImage = UIImage()

class RxImagePicker {
    init(owner: UIViewController) {

    }

    func pickImage() -> Single<UIImage> {
        return Single.just(expectedImage)
    }
}

class RxImagePickerTests: XCTestCase {

    let underTest = RxImagePicker(owner: UIViewController())
    var disposeBag = DisposeBag()

    func testCanPickImage() {
        var resultImage: UIImage?

        underTest.pickImage().subscribe(onSuccess: { (image) in
            resultImage = image
        }) { (_) in
            XCTFail()
        }.disposed(by: disposeBag)

        expect(resultImage).toEventuallyNot(beNil())
    }

    func testCompareImageInstance() {
        var resultImage: UIImage?

        underTest.pickImage().subscribe(onSuccess: { (image) in
            resultImage = image
        }) { (_) in
            XCTFail()
        }.disposed(by: disposeBag)

        expect(resultImage).toEventually(equal(expectedImage))
        expect(resultImage).toEventuallyNot(equal(notExpectedImage))
    }
}

```

중복을 제거 합니다.

```
func pickImage() -> Single<UIImage> {  
    return Single.create(subscribe: { [weak self] (single) -> Disposable in  
        let disposable = Disposables.create()  
  
        guard let `self` = self else {  
            single(.error(NSError(domain: "RxImagePicker Error", code: 0, userInfo: nil)))  
            return disposable  
        }  
  
        self.picker.showAlbum(with: self.owner, complete: { (images, canceled) in  
            if let pickedImage = images.first, !canceled {  
                single(.success(pickedImage))  
            }  
        })  
        return disposable  
    })  
}
```

```
func pickImage() -> Single<UIImage> {  
  
    return Single.create(subscribe: { [weak self] (single) -> Disposable in  
        let disposable = Disposables.create()  
  
        guard let `self` = self else {  
            single(.error(NSError(domain: "RxImagePicker Error", code: 0, userInfo: nil)))  
            return disposable  
        }  
  
        self.picker.showAlbum(with: self.owner, complete: { (images, canceled) in  
            if let pickedImage = images.first, !canceled {  
                single(.success(pickedImage))  
            }  
        })  
        return disposable  
    })  
})  
}
```



```
class RxImagePicker {
    let picker: UIImagePickerControllerAdapterType
    let owner: UIViewController

    init(owner: UIViewController, picker: UIImagePickerControllerAdapterType) {
        self.owner = owner
        self.picker = picker
    }

    func pickImage() -> Single<UIImage> {
        return Single.create(subscribe: { [weak self] (single) -> Disposable in
            let disposable = Disposables.create()

            guard let `self` = self else {
                single(.error(NSError(domain: "RxImagePicker Error", code: 0, userInfo: nil)))
                return disposable
            }

            self.picker.showAlbum(with: self.owner, complete: { (images, canceled) in
                if let pickedImage = images.first, !canceled {
                    single(.success(pickedImage))
                }
            })
            return disposable
        })
    }
}
```



```
 typealias PickImageComplete = (_ images: [UIImage], _ cancel: Bool) -> ()
 protocol MImagePickerRxAdapterType {
     func showAlbum(with: UIViewController, complete: PickImageComplete)
 }

 class MockAlbum {
     static let expectedImage = UIImage()
     static let notExpectedImage = UIImage()
 }

 class MockImagePickerRxAdapter: MImagePickerRxAdapterType {
     func showAlbum(with: UIViewController, complete: PickImageComplete) {
         complete([MockAlbum.expectedImage], false)
     }
 }
```

**원래 objc 클래스를 참조하여
complete block, protocol을 정의 합니다.**

```
 typealias PickImageComplte = (_ images: [UIImage], _ cancel: Bool) -> ()
 protocol MDImagePickerRxAdapterType {
     func showAlbum(with: UIViewController, complete: PickImageComplte)
 }

```

Swift Protocol

```
 typedef void (^ImagePickerCompleteBlock)(NSArray* images, BOOL isCanceled);

 @protocol MDImagePickerRxAdapterType
 - (void)showAlbumWith:(UIViewController *)viewController complete:(ImagePickerCompleteBlock)complete;
 @end

```

Objc Protocol

```
 @interface MDImagePickerController : NSObject <MDImagePickerRxAdapterType>

```

레거시 클래스에 부착

RxImagePicker

클래스파일을 만들고 메인 타겟으로
위치를 옮깁니다.

```
let imagePicker = RxImagePicker(picker: UIImagePickerController.instance())
```

```
return self.imagePicker.pickImage(owner: vc)  
    .flatMap { self.imageUploader.upload(image: $0) }  
    .asObservable()  
    .flatMap { Observable<Mutation>.just(.updateImageUrl(url: URL(string: $0))) }
```

RxImagePicker Default Value

```
let imagePicker = RxImagePicker(picker: UIImagePickerController.instance())
```

그런데 만약 TDD로 하지
않았다면?

TDD로 안 한 경우

Rx 이미지 피커를 붙일 대상을 만든 경우 패스

이미지 피커를 붙일 대상을 아직 못 만든 경우 일단 붙여 볼 버튼 고르기 주석 처리 후 실행 코드

앱실행 이미지 피커 실행 이미지 선택 이미지 로딩 기다림

이미지 업로드 기다림 이미지 업로드 완료

뭔가 변화가 생길 때 마다 반복

귀찮아서 안 하면 꼭 QA에 걸림

TDD로 하면..

아시다시피..

$\text{⌘} + U$

서버 API가
아직 준비되지 않음

뭉쳐 있으면 어렵다.

MVC

Massive VC

Massive VC

Massive VC

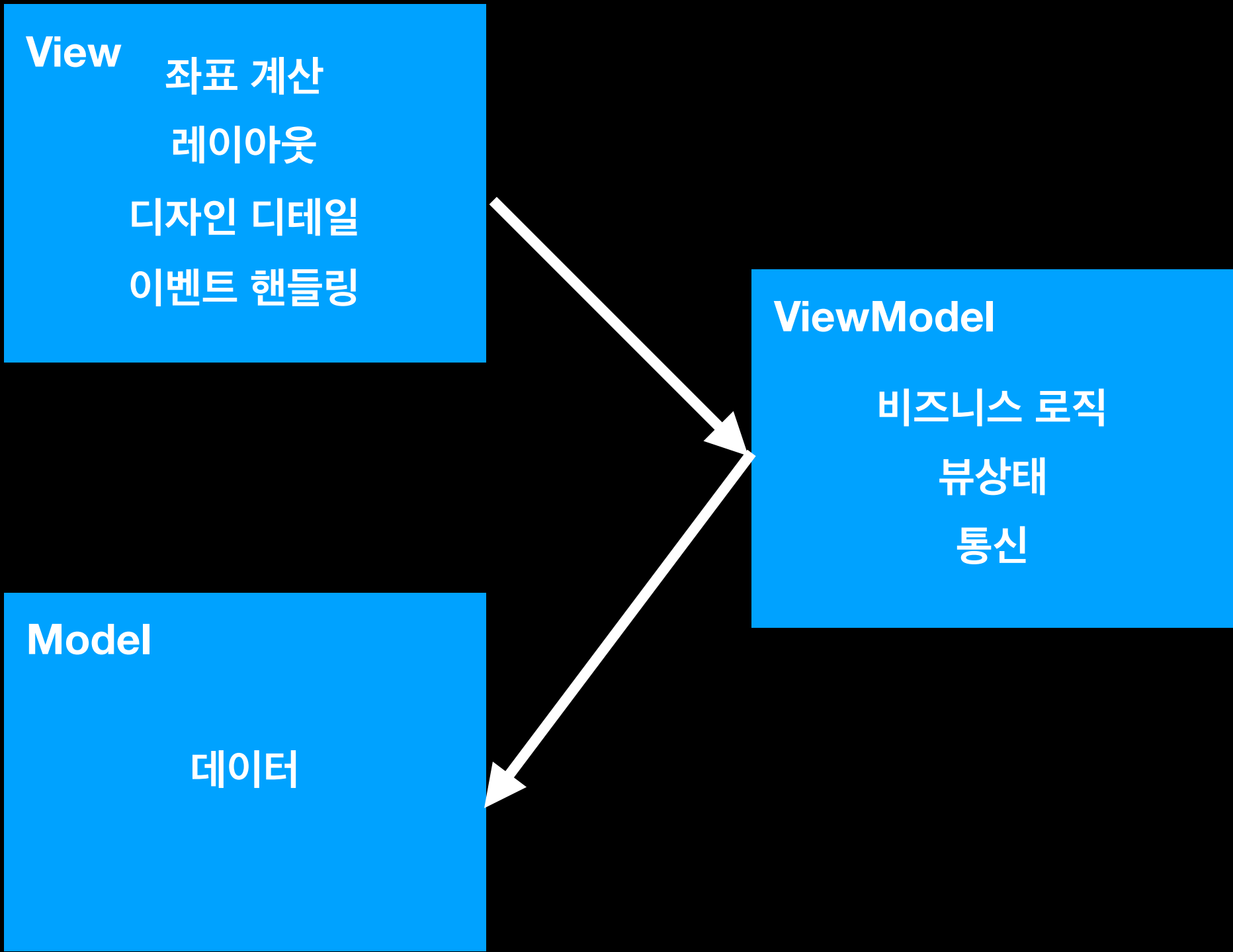
Massive VC

뷰상태
데이터
비즈니스 로직
좌표 계산
레이아웃
디자인 디테일
통신
이벤트 핸들링

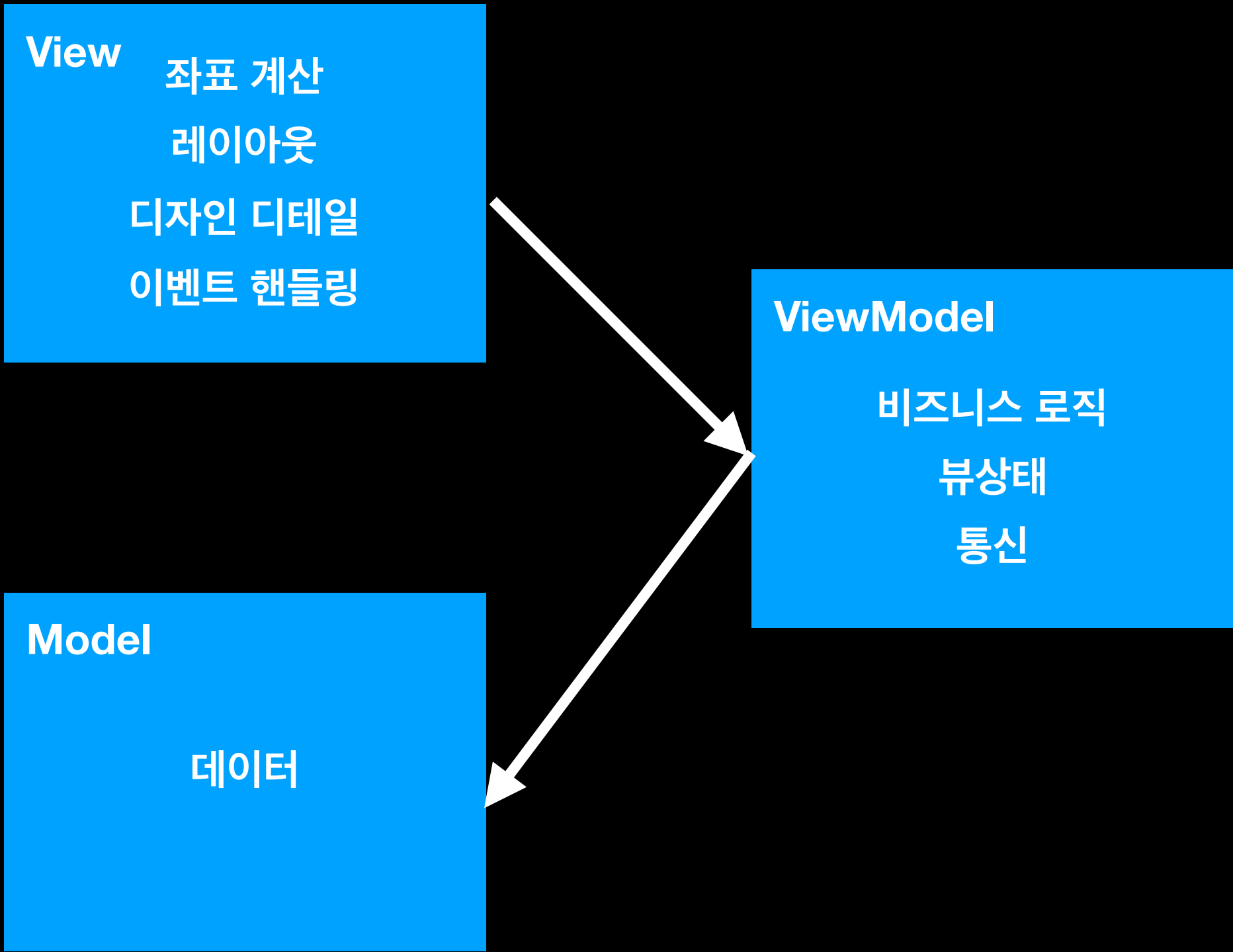
MVVM 도입!!

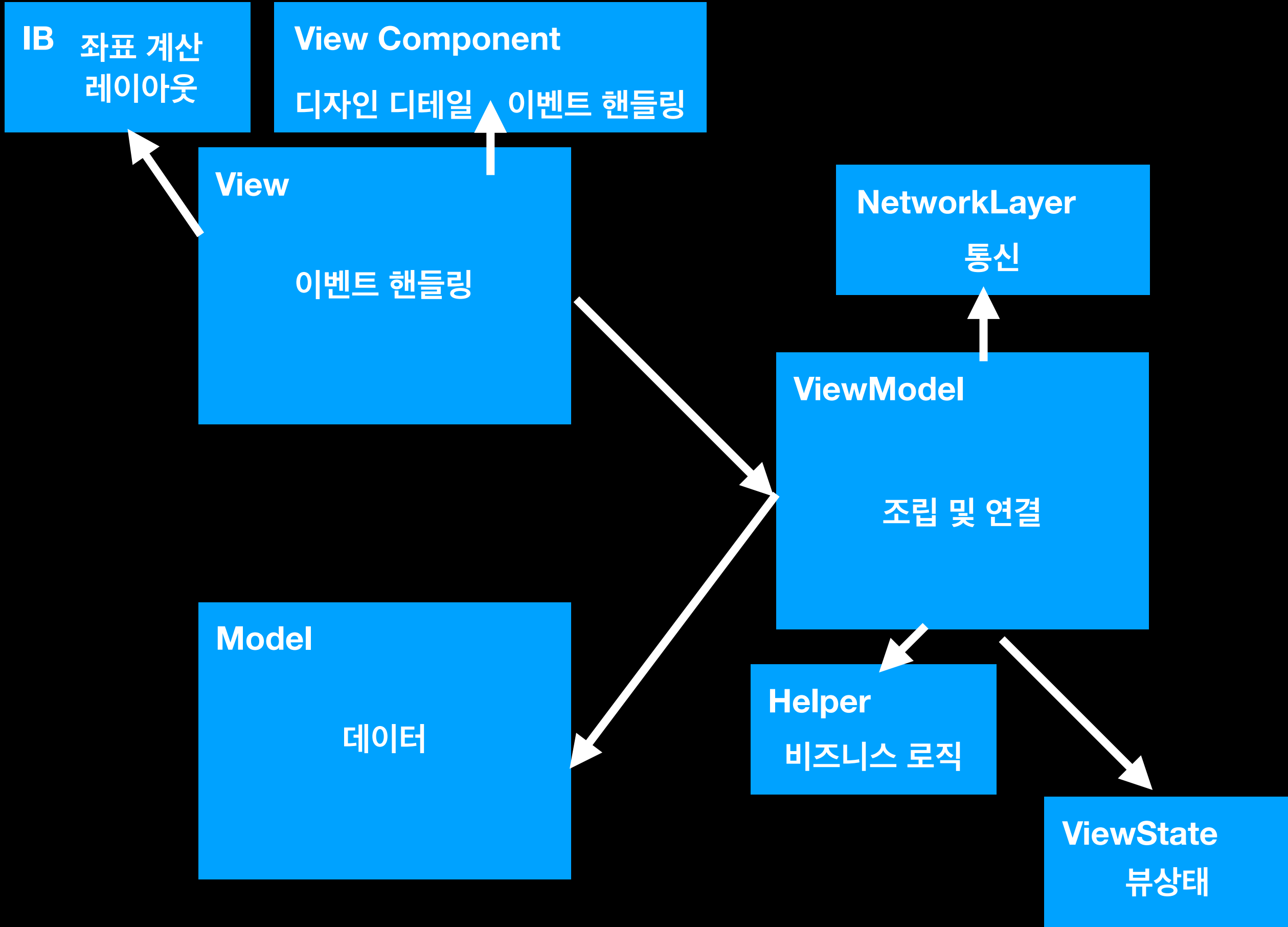
Massive VC

뷰상태
데이터
비즈니스 로직
좌표 계산
레이아웃
디자인 디테일
통신
이벤트 핸들링

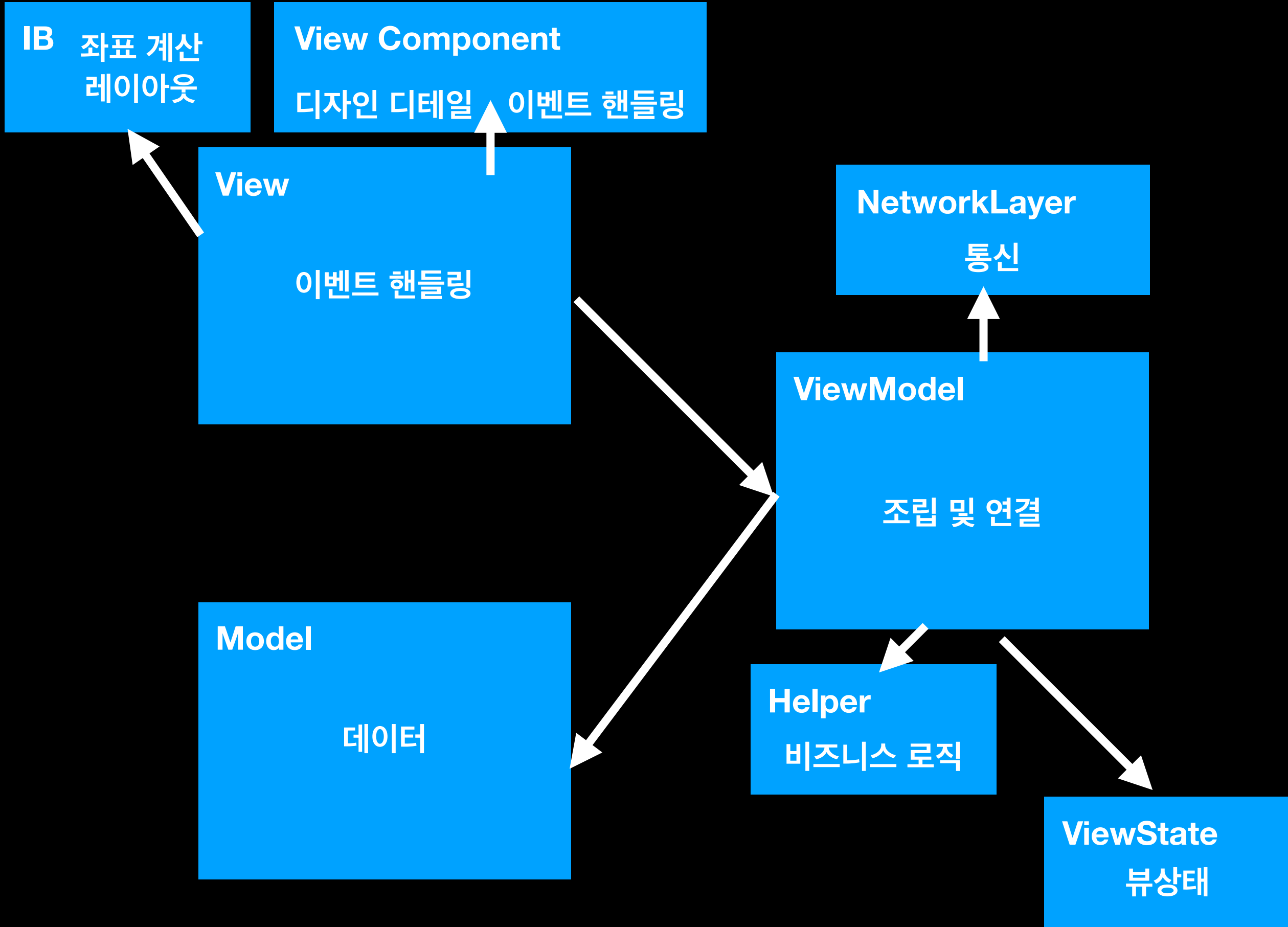


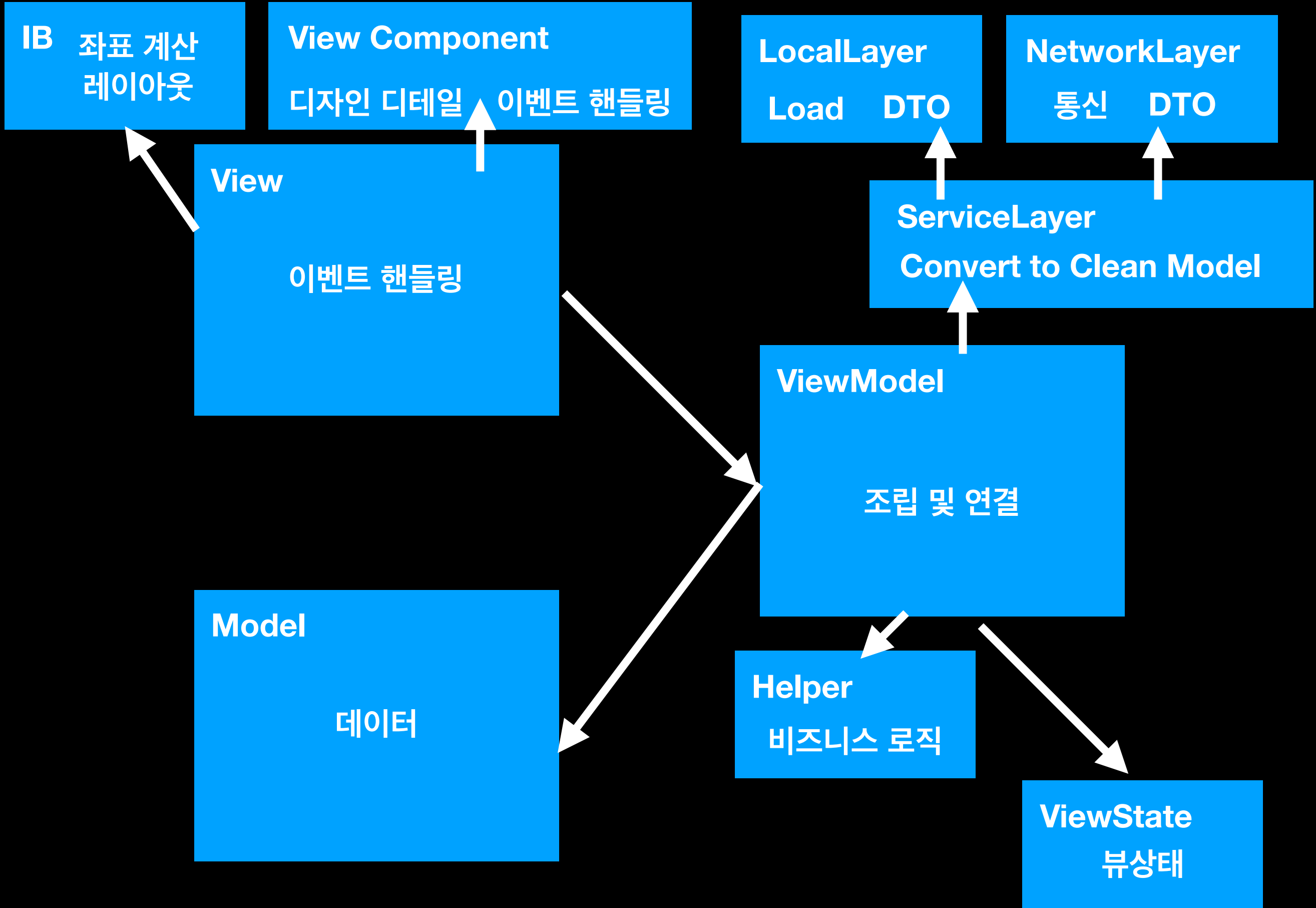
책임 나누기

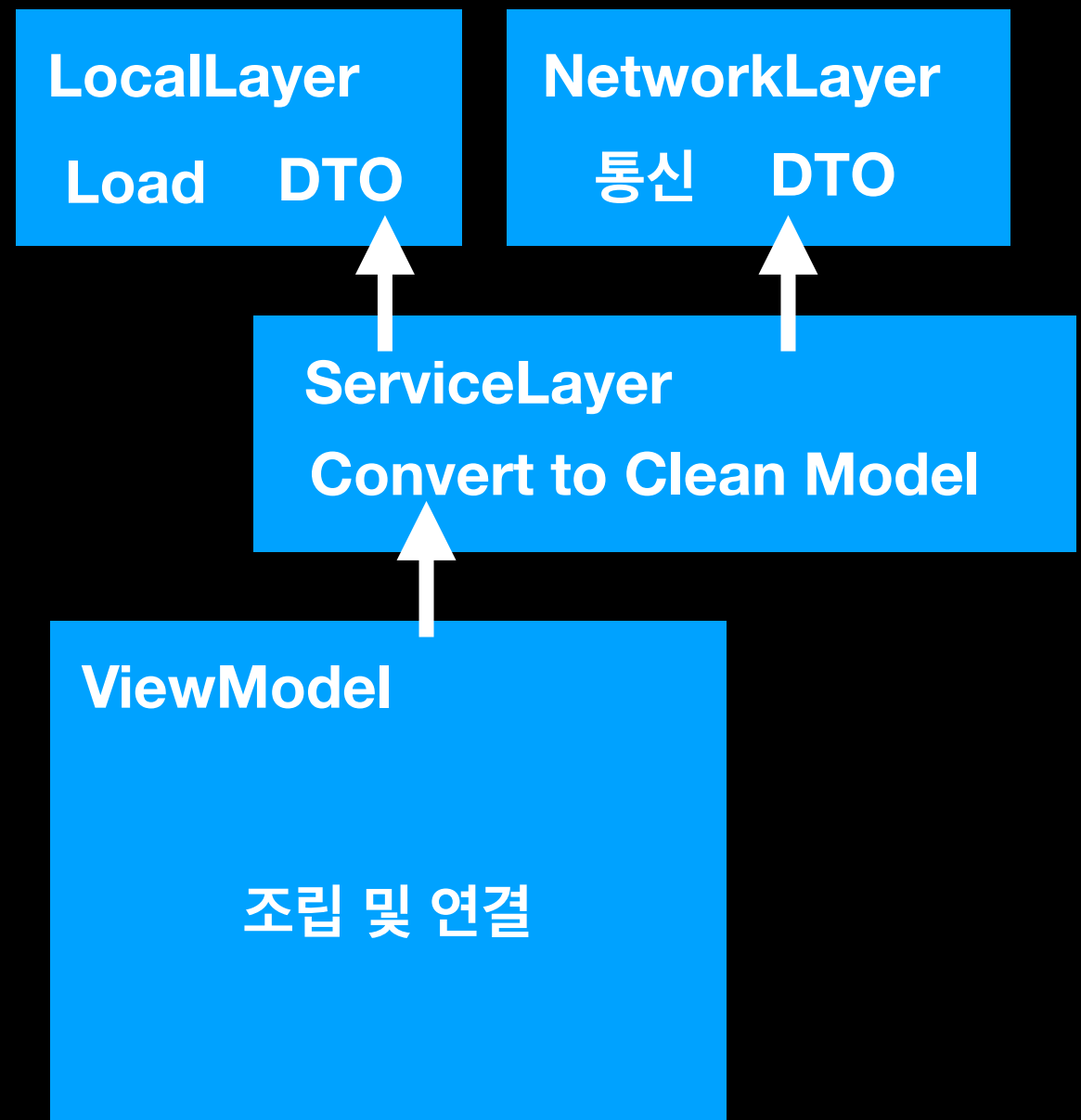




DTO를 이용한 외부 의존성 분리



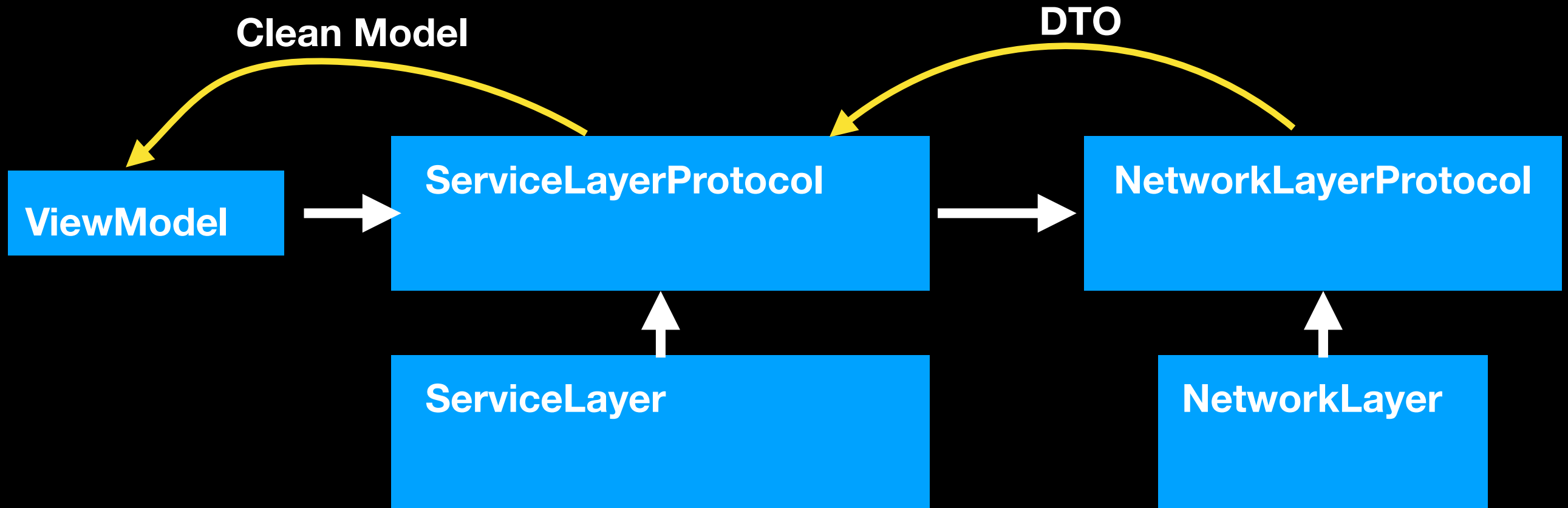






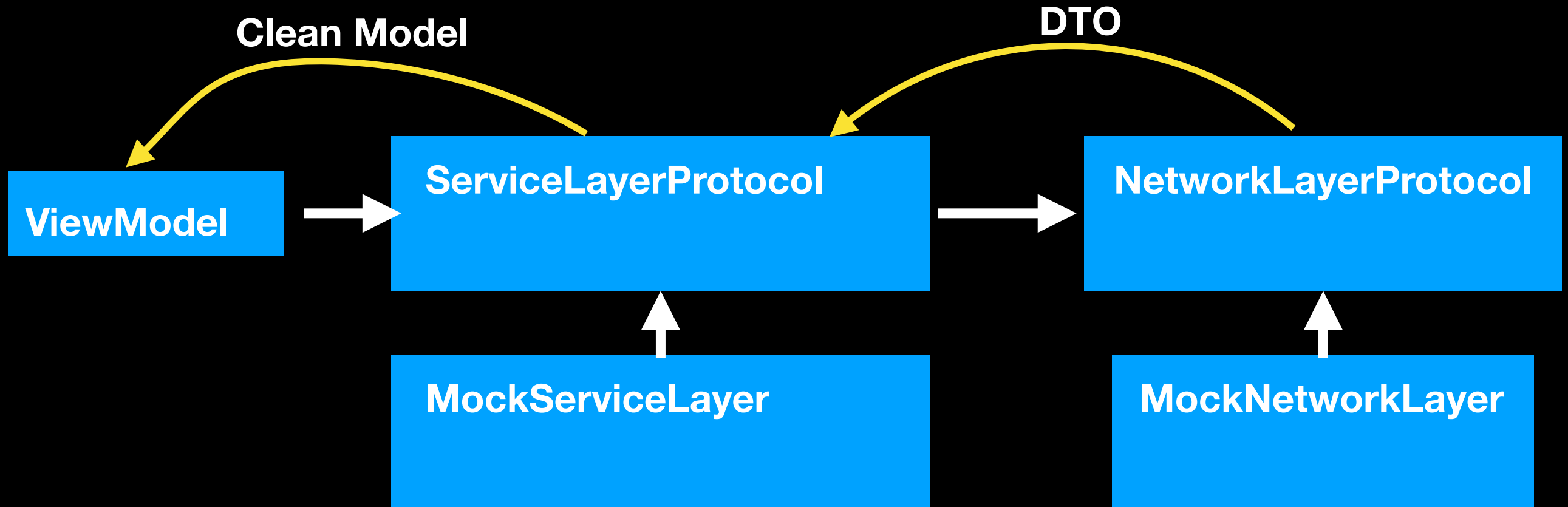
스펙에 명시되어 있고,
iOS 개발 컨벤션에 알맞고
우리가 개발하기 좋은 추상화 레벨을 갖춘

서버에서 줄 것 같은 형식
틀려도 됨
Codable



스펙에 명시되어 있고,
iOS 개발 컨벤션에 알맞고
우리가 개발하기 좋은 추상화 레벨을 갖춘

서버에서 줄 것 같은 형식
틀려도 됨
Codable



UI라서
테스트하기 어렵다
고 생각되지만
실제로는 아닌 것들

좌표 계산
상태 변화

애니메이션
드래그 앤 드롭

라이브러리 학습 테스트

SwiftDate

Locale, TimeZone을 관리하고 변환하는 기능 제공 라이브러리


```
func testReadableDateString() {
    let basisDate: DateInRegion = "2018-04-01T08:00+0900".date(format: .iso8601Auto, fromRegion: Region(tz: TimeZone(identifier: "Asia/Seoul")!, loc:
        LocaleName.korean.locale))!

    let df = DateFormatter()
    df.locale = LocaleName.korean.locale

    df.setLocalizedDateFormatFromTemplate("yyyy MMM dd")
    expect(df.string(from: basisDate.absoluteDate)).to(equal("2018년 4월 01일"))

    df.setLocalizedDateFormatFromTemplate("yyyy MM dd")
    expect(df.string(from: basisDate.absoluteDate)).to(equal("2018. 04. 01. "))

    df.locale = LocaleName.englishUnitedStates.locale

    df.setLocalizedDateFormatFromTemplate("yyyy MMM dd")
    expect(df.string(from: basisDate.absoluteDate)).to(equal("Apr 01, 2018"))

    df.setLocalizedDateFormatFromTemplate("yyyy MM dd")
    expect(df.string(from: basisDate.absoluteDate)).to(equal("04/01/2018"))
}
```

```
func testReadableTimeString() {
    let basisDate: DateInRegion = "2018-04-01T08:00+0900".date(format: .iso8601Auto, fromRegion: Region(tz: TimeZone(identifier: "Asia/Seoul")!, loc:
        LocaleName.korean.locale))!

    let df = DateFormatter()
    df.locale = LocaleName.korean.locale

    df.timeStyle = .full
    expect(df.string(from: basisDate.absoluteDate)).to(equal("오전 8시 0분 0초 대한민국 표준시"))

    df.setLocalizedDateFormatFromTemplate("HHmm")
    expect(df.string(from: basisDate.absoluteDate)).to(equal("08:00"))

    df.timeStyle = .long
    expect(df.string(from: basisDate.absoluteDate)).to(equal("오전 8시 0분 0초 GMT+9"))

    df.timeStyle = .medium
    expect(df.string(from: basisDate.absoluteDate)).to(equal("오전 8:00:00"))

    df.timeStyle = .short
    expect(df.string(from: basisDate.absoluteDate)).to(equal("오전 8:00"))

    df.locale = LocaleName.englishUnitedStates.locale

    df.timeStyle = .full
    expect(df.string(from: basisDate.absoluteDate)).to(equal("8:00:00 AM Korean Standard Time"))

    df.timeStyle = .long
    expect(df.string(from: basisDate.absoluteDate)).to(equal("8:00:00 AM GMT+9"))

    df.timeStyle = .medium
    expect(df.string(from: basisDate.absoluteDate)).to(equal("8:00:00 AM"))

    df.timeStyle = .short
    expect(df.string(from: basisDate.absoluteDate)).to(equal("8:00 AM"))
}
```

```
let df = DateFormatter()
df.locale = LocaleName.korean.locale
df.setLocalizedDateFormatFromTemplate("E")
expect(df.string(from: basisDate.absoluteDate)).to(equal("일"))

df.setLocalizedDateFormatFromTemplate("EE")
expect(df.string(from: basisDate.absoluteDate)).to(equal("일"))

df.setLocalizedDateFormatFromTemplate("EEE")
expect(df.string(from: basisDate.absoluteDate)).to(equal("일"))

df.setLocalizedDateFormatFromTemplate("EEEE")
expect(df.string(from: basisDate.absoluteDate)).to(equal("일요일"))

df.locale = LocaleName.englishUnitedStates.locale
df.setLocalizedDateFormatFromTemplate("E")
expect(df.string(from: basisDate.absoluteDate)).to(equal("Sun"))

df.setLocalizedDateFormatFromTemplate("EE")
expect(df.string(from: basisDate.absoluteDate)).to(equal("Sun"))

df.setLocalizedDateFormatFromTemplate("EEE")
expect(df.string(from: basisDate.absoluteDate)).to(equal("Sun"))

df.setLocalizedDateFormatFromTemplate("EEEE")
expect(df.string(from: basisDate.absoluteDate)).to(equal("Sunday"))
}
```

```
func testCreateMonthWithFirstDayOfMonth() {
    let underTest = "2018-04-01".date(format: .iso8601Auto)!

    expect(underTest.year).to(equal(2018))
    expect(underTest.month).to(equal(4))
}

func testCreateMonthWithNotFirstDayOfMonth() {
    let underTest = "2018-04-15".date(format: .iso8601Auto)!

    expect(underTest.year).to(equal(2018))
    expect(underTest.month).to(equal(4))
}

func testGetNextMonth() {
    let underTest = "2018-04-15".date(format: .iso8601Auto)!
    let nextMonth = underTest.nextMonth
    expect(nextMonth.year).to(equal(2018))
    expect(nextMonth.month).to(equal(5))
}

func testParsingDateWithoutOffset() {
    let date = "2018-05-05T08:32+0900".date(format: .iso8601Auto, fromRegion: Region(tz: TimeZoneName.asiaSeoul.timeZone, loc: Locale.current))

    expect(TimeZoneName.asiaSeoul.timeZone.abbreviation()).to(equal("GMT+9"))
    expect(TimeZoneName.asiaSeoul.timeZone.description).to(equal("Asia/Seoul (current)"))

    expect(date).toNot(beNil())
    expect(date?.year).to(equal(2018))
    expect(date?.region.timeZone).toNot(equal(TimeZoneName.asiaSeoul.timeZone))
}
```

함께 해보기

[https://github.com/
vbmania/](https://github.com/vbmania/CountdownTimerTddExam)

CountdownTimerTddExam

TDD 3단계

TDD 3단계

TDD 3단계

1. 실패하는 테스트

TDD 3단계

1. 실패하는 테스트

2. 가장 빨리 성공하게

TDD 3단계

1. 실패하는 테스트

2. 가장 빨리 성공하게

3. 리팩토링

TDD 3단계

1. 실패하는 테스트

2. 가장 빨리 성공하게

3. 리팩토링

3-1. 중복을 제거하고

TDD 3단계

1. 실패하는 테스트

2. 가장 빨리 성공하게

3. 리팩토링

3-1. 중복을 제거하고

3-2. 의미를 드러낸다.

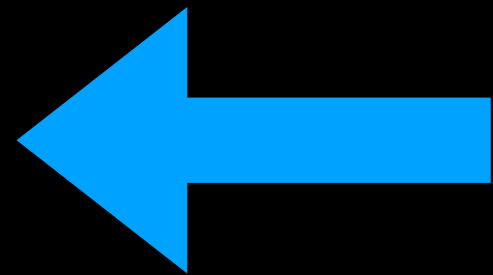
(명확하게 한다.)

TDD 3단계

1. 실패하는 테스트

2. 가장 빨리 성공하게

3. 리팩토링



3-1. 중복을 제거하고

3-2. 의미를 드러낸다.

(명확하게 한다.)

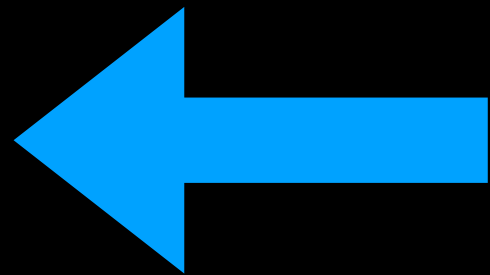
TDD 3단계

1. 실패하는 테스트

2. 가장 빨리 성공하게

3. 리팩토링

상수중복
의미중복



3-1. 중복을 제거하고

3-2. 의미를 드러낸다.

(명확하게 한다.)

시연

[https://github.com/
vbmania/](https://github.com/vbmania/CountdownTimerTddExam)

CountdownTimerTddExam

TDD의 한계

Silver Bullet은 아님

테스트 한 만큼만 보장됨

Q&A

- 흥미로운 질문입니다.
- 저도 한번 고민해봐야 겠네요.
- 노력하면 가능하지 않을까요?
- 아.. 거기까지는 생각해보지 못했습니다.
- 그건 사실이 아닙니다.
- 안타깝지만 그건... 좀...
- 대답 드리기에 시간이 부족하군요...

참고자료

- **Test Driven Development: By Example - 캔트 백**
- **Effective Unit Testing - 개발자를 위한 단위 테스트**
- **테스트 주도 개발 - 고품질 쾌속 개발을 위한 TDD 실천법과 도구**
- **WWDC 2017**
 - **409 Whats New in Testing**
 - **414 Engineering for Testability**
- **WWDC 2018**
 - **403 Whats New in Testing**
 - **417 Testing Tips & Tricks**
- **기타 등등 블로그 아티클들..**
- **내 삽질**