Fast campus

Delegate

Giftbot

Delegate

Fast campus

영어사전

delegate 미국식 동사 ['delɪgeɪt] 영국식 ★ 다른 뜻(2건) | 예문보기

1. (집단의 의사를 대표하는) 대표(자) 2. (권한업무 등을) 위임하다 3. (대표를) 뽑다

특정 로직을 내가 아닌 다른 객체가 대신 구현하도록 위임하는 형태의 디자인 패턴

요청하는 객체와 요청에 응답할 객체로 나누어 작성

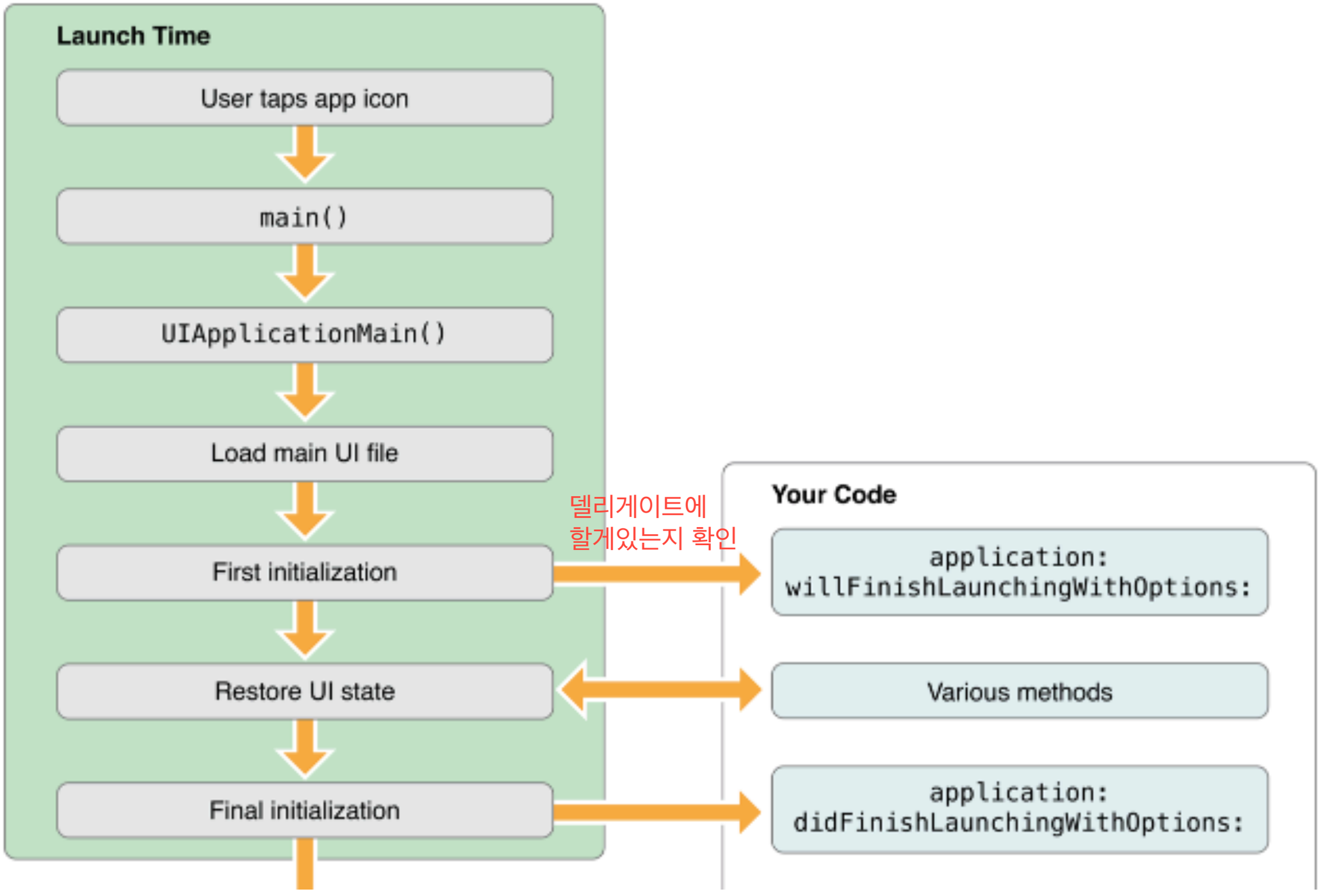
주로 다음과 같은 상황에 사용

- 뷰가 받은 이벤트나 상태를 뷰컨트롤러에게 전달하여 처리 (View -> ViewController)

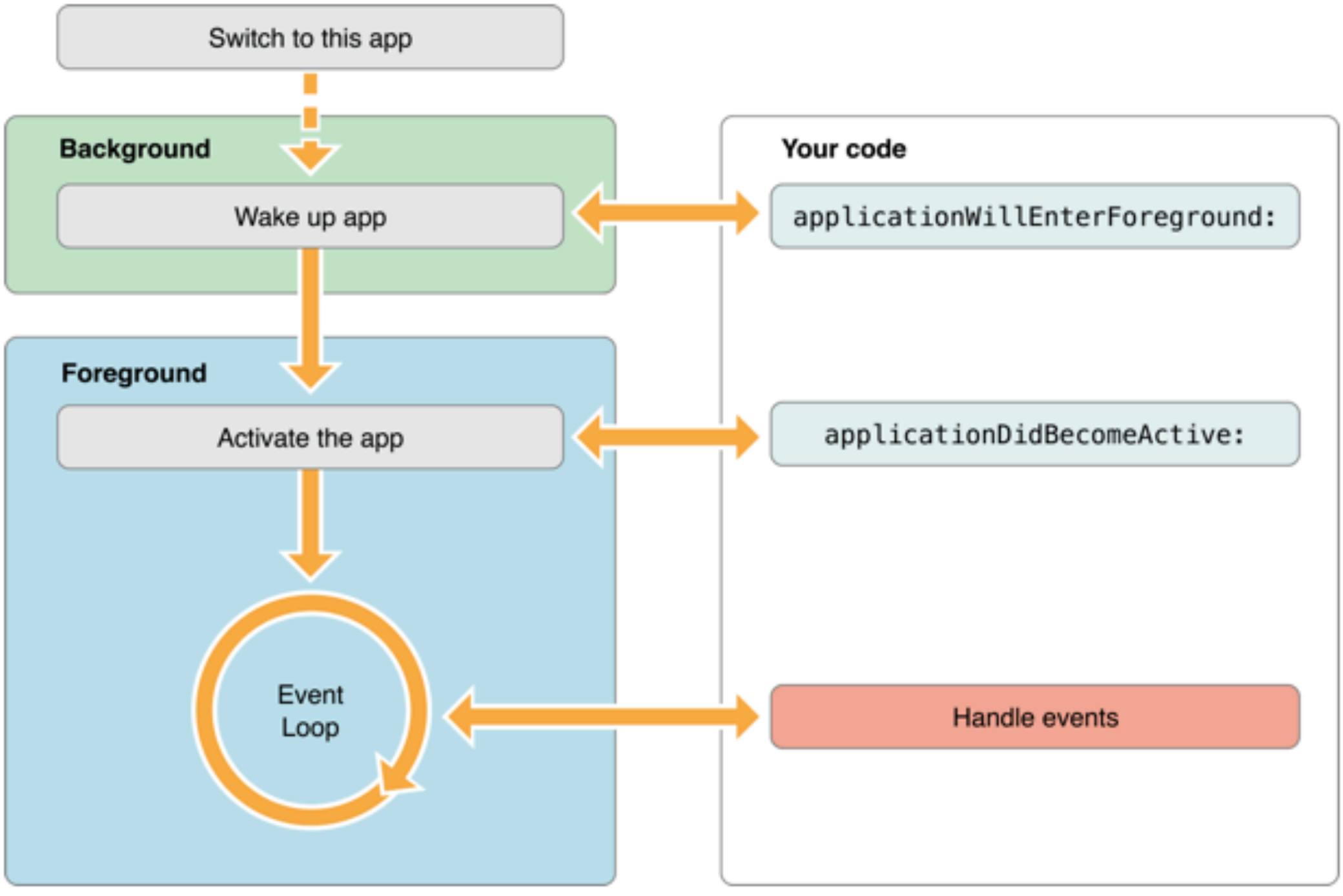
- 뷰 구성에 필요한 정보를 뷰컨트롤러가 결정 (View <- ViewController)

- 주요 코드는 숨기고 특정 상황에 대해서만 커스터마이징 할 수 있도록 제공

AppDelegate



AppDelegate



Delegate protocol

```
protocol CustomViewDelegate: class {
   func colorForBackground(_ newColor: UIColor?) -> UIColor
}
```

배경색이 변할 때 해당 함수를 커스텀 클래스에 넘겨줘서 실행하게 해준다.

Delegate 선언

```
final class CustomView: UIView {
  weak var delegate: CustomViewDelegate?

  override var backgroundColor: UIColor? {
    get { return super.backgroundColor }
    set {
      let color = delegate?.colorForBackground(newValue)
      let newColor = color ?? newValue ?? .gray
      super.backgroundColor = newColor
      print("새로 변경될 색은 :", newColor)
    }
  }
}
```

Delegate 구현부

```
class ViewController: UIViewController, CustomViewDelegate {
  @IBOutlet weak var customView: CustomView!

  override func viewDidLoad() {
    super.viewDidLoad()
    customView.delegate = self
  }

  func colorForBackground(_ newColor: UIColor?) -> UIColor {
    guard let color = newColor else { return .gray }
    return color == .green ? .blue : color
  }
}
```

Delegate

< 위임하는 객체 >

```
protocol CustomViewDelegate: class {
  func colorForBackground(_ newColor: UIColor?) -> UIColor
}


final class CustomView: UIView {
  weak var delegate: CustomViewDelegate?

  override var backgroundColor: UIColor? {
    get { return super.backgroundColor }
    set {
      let color = delegate?.colorForBackground(newValue)
      let newColor = color ?? newValue ?? .gray
      super.backgroundColor = newColor
      print("새로 변경될 색은 :", newColor)
    }
  }
}
```

< 위임받아 처리하는 객체 >

```
class ViewController: UIViewController, CustomViewDelegate {
  @IBOutlet weak var customView: CustomView!

  override func viewDidLoad() {
    super.viewDidLoad()
    customView.delegate = self
  }

  func colorForBackground(_ newColor: UIColor?) -> UIColor {
    guard let color = newColor else { return .gray }
    return color == .green ? .blue : color
  }
}
```

Delegate

< 위임하는 객체 >

1

```
protocol CustomViewDelegate: class {
  func colorForBackground(_ newColor: UIColor?) -> UIColor
}
```

```
final class CustomView: UIView {
  weak var delegate: CustomViewDelegate?
   delegate = self 에서 순환참조가 일어날 수 있기 때문
  override var backgroundColor: UIColor? {
    get { return super.backgroundColor }
    set {
      let color = delegate?.colorForBackground(newValue)
      let newColor = color ?? newValue ?? .gray
      super.backgroundColor = newColor
      print("새로 변경될 색은 :", newColor)
    }
  }
}
```

2

3

아래에서 순환참조가 발생하게 되는 상황
1. <위임받아 처리하는 객체> 에서 @IBOutlet이 weak으로 선언되지 않고
2. <위임하는 객체>에서 delegate를 weak으로 선언하지 않는 경우

< 위임받아 처리하는 객체 >

1. 프로토콜 정의

```
class ViewController: UIViewController, CustomViewDelegate {
  @IBOutlet weak var customView: CustomView!
  override func viewDidLoad() {
    super.viewDidLoad()
    customView.delegate = self
  }
  func colorForBackground(_ newColor: UIColor?) -> UIColor {
    guard let color = newColor else { return .gray }
    return color == .green ? .blue : color
  }
}
```

2. delegate 프로퍼티 선언
   - 일반적으로 delegate 라고 명명
   - 타입은 정의한 프로토콜과 동일
   - optional
   - weak
3. 필요한 곳에서 delegate 객체의 메서드 실행
   - delegate에 할당된 객체는 없을 수 있음 (nil)
   - nil이 아니면 메서드 호출에 응답하여
     어떤 로직(그게 무엇이 될지는 알 수 없음)을 수행
   - 반환되는 값이 있을 경우 그 결과를 받아서 활용

특히, <위임받아 처리하는 객체>에서 우리가 IBOutlet으로 만든 customVeiw를 깡코딩으로 할 경우
weak을 안쓰는 경우가 많은데 이 때 조심해야함. 순환참조가 발생함.

Delegate

< 위임하는 객체 >

1. delegate 프로토콜 채택
2. delegate 프로퍼티를 소유한 객체에게 위임받아 처리할 메서드를 구현한 객체 할당 (보통 self)
   - 이 부분을 실수하는 경우가 많음
   - 프로토콜을 채택하지 않은 경우 에러 메시지 발생
3. 채택한 프로토콜의 메서드 구현
   - 옵셔널 메서드인 경우 미구현 가능
   - 이 메서드가 언제 호출될 지는 결정할 수 없으며, 그 시점은 위임하는 객체에 달려 있음

< 위임받아 처리하는 객체 >

1
2
3

```swift
class ViewController: UIViewController, CustomViewDelegate {
  @IBOutlet weak var customView: CustomView!

  override func viewDidLoad() {
    super.viewDidLoad()
    customView.delegate = self
  }

  func colorForBackground(_ newColor: UIColor?) -> UIColor {
    guard let color = newColor else { return .gray }
    return color == .green ? .blue : color
  }
}
```