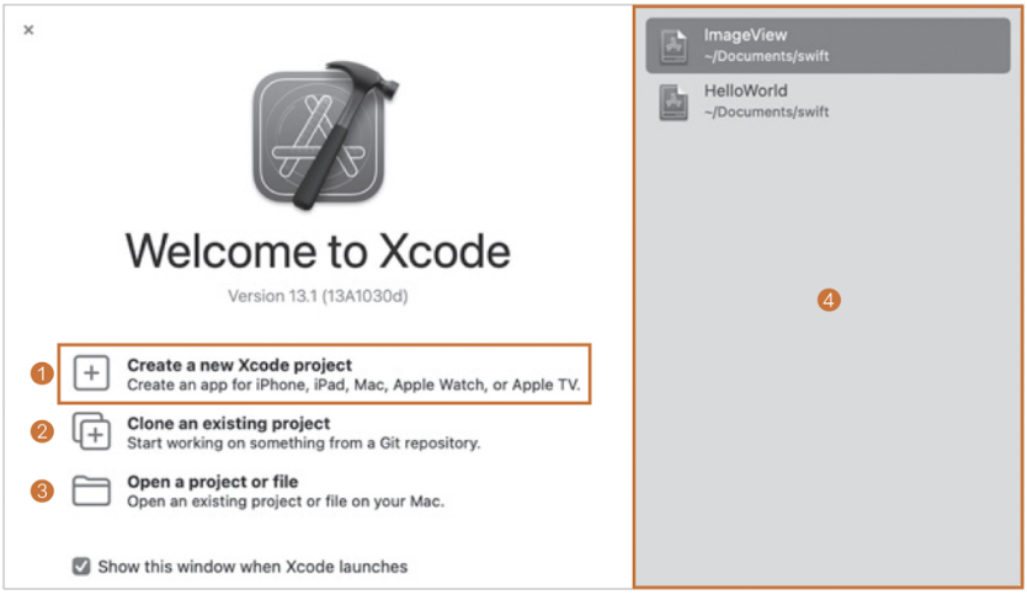
02. Hello World 앱 만들며 Xcode에 완벽 적응하기

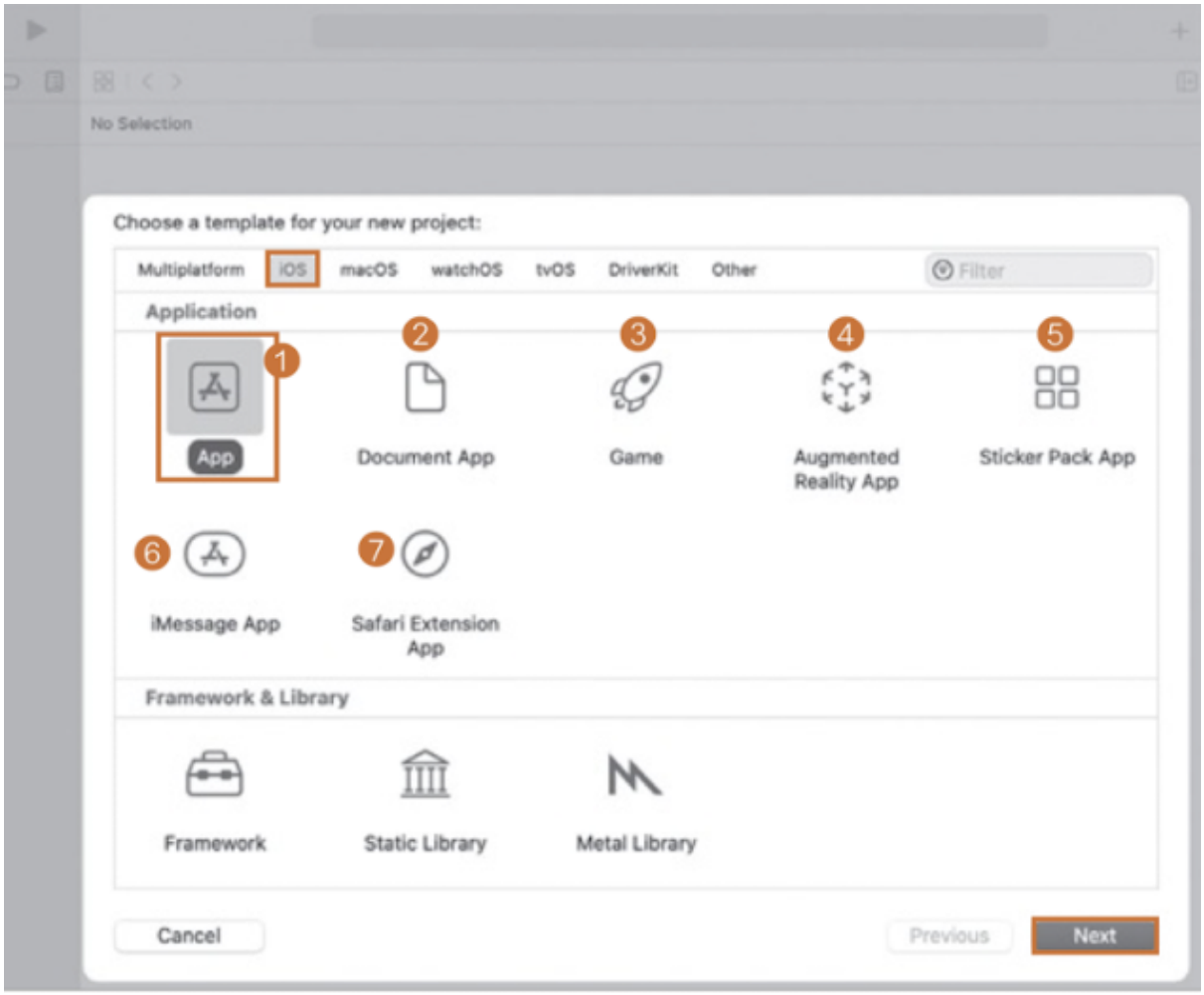
02-1 새 프로젝트 시작하기

1. Xcode 실행하기



- Create a new ...
  - 아이폰, 아이패드, 맥 앱을 만들기 위한 새로운 Xcode 프로젝트 생성
- Clone as existing project
  - SVN이나 git과 같은 버전 관리 도구로 연결하여 기존 소스를 가져올 수 있다

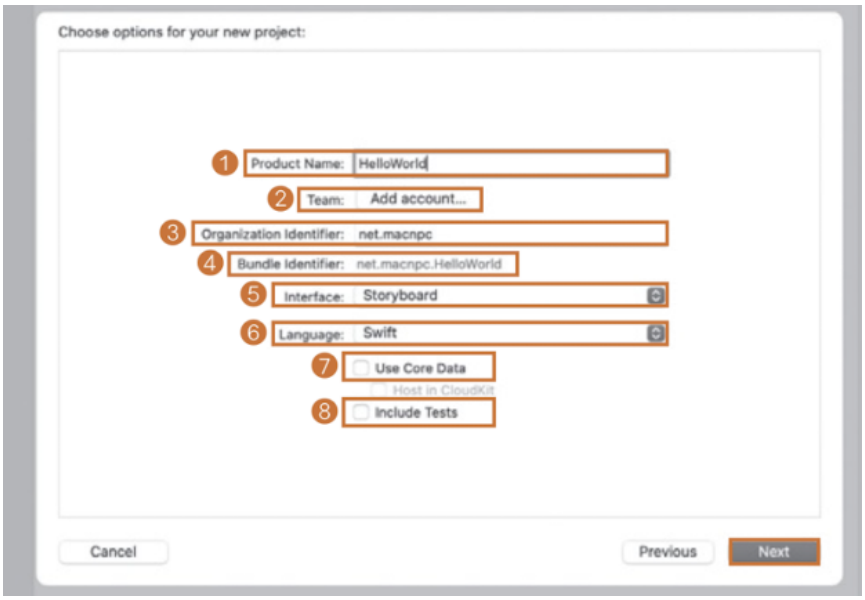
2. 템플릿 선택하기



- App
  - 뷰를 사용하는 앱을 개발할 때 사용하는 템플릿
  - 기본적으로 하나의 뷰가 나타나며 필요에 따라 새로운 뷰를 추가 가능
- Document App
  - 데이터를 저장할 수 있는 문서기반의 앱을 개발할 때 사용
- Game
  - 게임 앱을 개발할 때 사용하는 템플릿
  - 그래픽 처리를 위한 OpenGL 게임 뷰를 생성해 준다
- Augmented Reality App
  - 증강현실 앱을 개발할 때 사용
- Sticker Pack App
  - 스티커 팩 앱을 개발할 때 사용

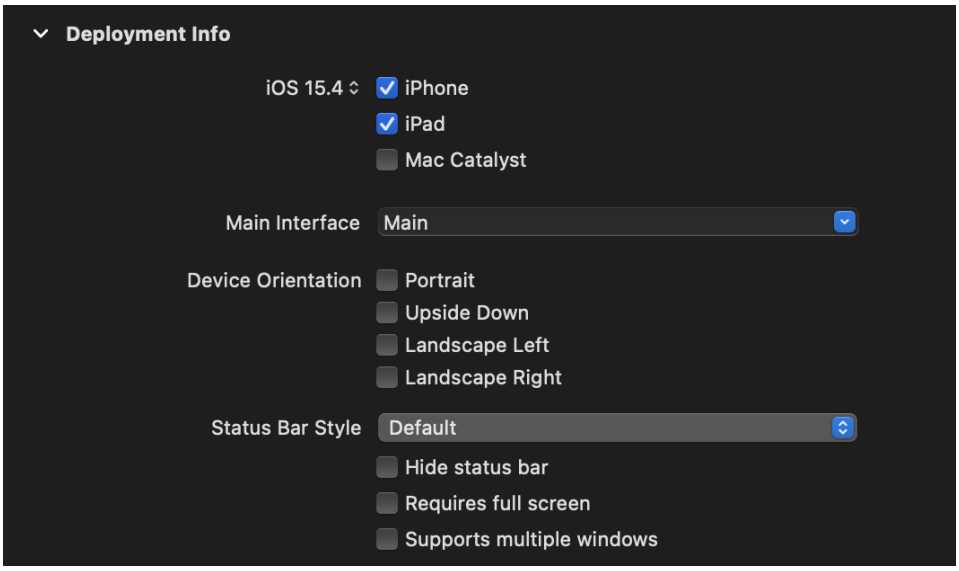
- iMessage App
  - 아이메시지 앱을 개발할 때 사용
- Safari Extension App
  - 사파리 확장 앱을 개발할 때 사용하는 템플릿

3. 프로젝트의 기본 정보 입력



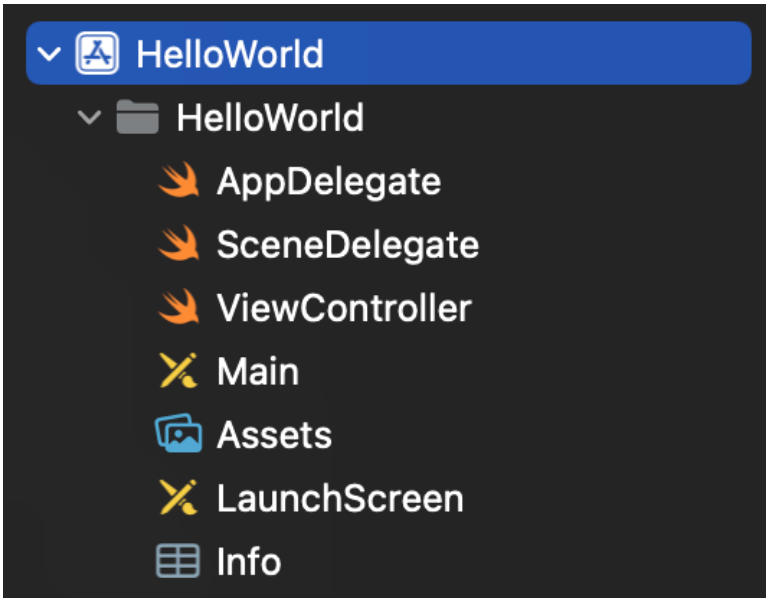
- Product Name
  - 개발하려고 하는 앱의 이름
- TEAM
  - 개발자 프로그램에 등록된 ID 또는 팀을 입력
  - 개발자 인증서가 등록되어 있으면 여기서 선택 가능
- Organization Identifier
  - 조직의 식별자를 입력
  - 일반적으로 개인이나 조직의 도메인 주소(URL)를 역순으로 입력
- Bundle Identifier
  - 식별자. ‘Company.Identifier.Product Name’으로 자동 생성된다.
  - 앱을 앱스토어에 등록할 때 다른 앱들과 구분하는 용도로 사용

- Interface
  - 스토리보드(Storyboard)와 스위프트UI(Swift UI)중에 하나를 선택할 수 있다.



- Devices
  - 앱이 지원하는 기기를 선택 가능
- Device Orientation
  - 앱이 지원할 회전 방향 선택 가능
  - 세워진 상태(Protrait) / 거꾸로 뒤집어진 상태(Upside Down)
    왼쪽으로 회전(Landscape Left) / 오른쪽으로 회전(Landscape Right)

프로젝트 파일들 살펴보기



- AppDelegate.swift
  - 앱의 실행주기(Life Cycle)를 관리하는 내용의 스위프트 소스 코드가 들어 있는 클래스 파일
  - 앱을 실행하거나 종료 또는 백그라운드를 실행할 때 하는 일들을 관리
  - 일반적으로 초보 단계일 때는 프로그래머가 직접 코딩하지 않아도 된다
- SceneDelegate.swift
  - 사용자 인터페이스(User Interface:UI)의 실행주기(Life Cycle)를 관리하는 내용의 스위프트 소스 코드가 들어있는 클래스 파일
  - AppDelegate와 마찬가지로 초보 단계일 때는 프로그래머가 직접 코딩하지 않음
- ViewController.swift
  - 화면에서 보이는 뷰에서 처리하는 내용의 스위프트 소스 코드를 담고 있는 클래스 파일
  - 일반적으로 프로그래머는 이 파일에서 코딩을 하게 되며 뷰 하나당 클래스 하나가 대응
  - 스토리보드에서 여러 개의 뷰를 추가하면 뷰의 개수만큼 뷰 컨트롤러 클래스 파일 필요
- Main.storyboard
  - 앱의 내용을 시각적으로 쉽게 이해하고 프로그래밍 할 수 있도록 그림으로 표현한 파일

    - 화면에 보이는 내용 및 뷰와 뷰 간의 연결 관계 등을 표현할 수 있다
- Assets.xcassets
    - 앱의 아이콘을 보관하는 저장소
    - 이곳에서 앱 아이콘을 설정해야 원하는 앱 아이콘으로 표시 가능
- LaunchScreen.storyboard
    - 앱이 실행될 때 잠시 나타나는 스플래시 화면을 만드는 스토리보드
- Info.plist
    - 앱이 실행되는 데 필요한 정보를 저장하고 있는 파일

Xcode의 화면 구성 살펴보기

- 내비게이터 영역
    - 프로젝트 내비게이터, 심벌 내비게이터, 검색 내비게이터, 이슈 내비게이터
      테스트 내비게이터, 디버그 내비게이터, 브레이크 포인트 내비게이터
      리포트 내비게이터 등의 정보를 나타내 주는 영역

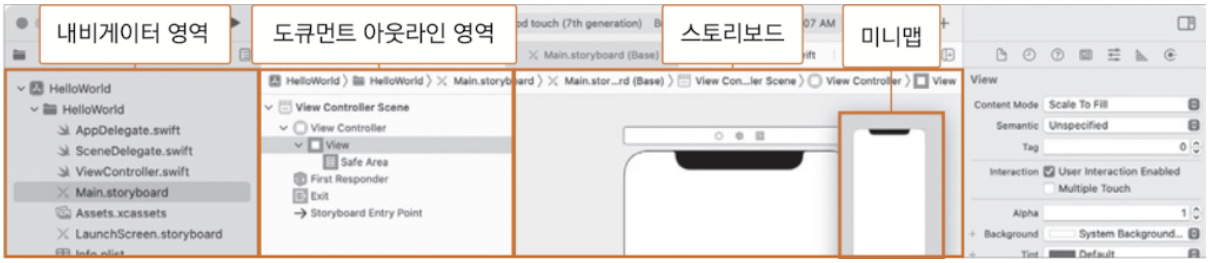
- 편집기 영역
  - 소스 파일을 열어 소스를 직접 입력하거나 스토리보드를 이용하여 화면을 디자인
- 인스펙터 영역
  - 소토리보드를 편집할 때 버튼, 컨트롤러, 뷰 등 모든 객체의 속성을 편집할 수 있는 영역
- 디버그 영역
  - 버그를 찾아 수정하는 과정인 디버그를 진행할 때 원하는 변수의 값을 확인하거나 테스트할 목적으로 사용한 입출력 내용이 출력되는 영역
  - 왼쪽의 변수 영역과 오른쪽의 콘솔 영역으로 구성되어 있다

02-2 스토리보드로 작업하기 위한 기본 환경 구성하기

스토리보드란?

- 예전에는 Xcode에서 화면을 구성할 때 인터페이스 빌더를 사용함
- 그런데 인터페이스 빌더는 각 화면 간의 연계성 및 흐름을 파악하기가 어려웠기 때문에 Xcode는 4.2버전부터
  스토리보드(Storyboard)라는 시각적인 기능을 제공하기 시작
- 앱의 화면 구성을 시각적이고 직관적으로 구성할 수 있게 지원하는 기능
- Xcode에서 만드록자 하는 앱이 어떤 모양으로 화면에 구성되어 있고, 버튼을 누르거나 화면을 스와이프하는 등의 특정 액션을 취했을 때 어떤 방식으로 화면 간 전환이 이루어지는지 보여줌
  화면 간의 흐름 및 전체적인 모양을 시각적인 방식으로 연결하고 표현해 줌으로써 직관적으로 앱의 흐름 확인

스토리보드 작업 환경 조정하기

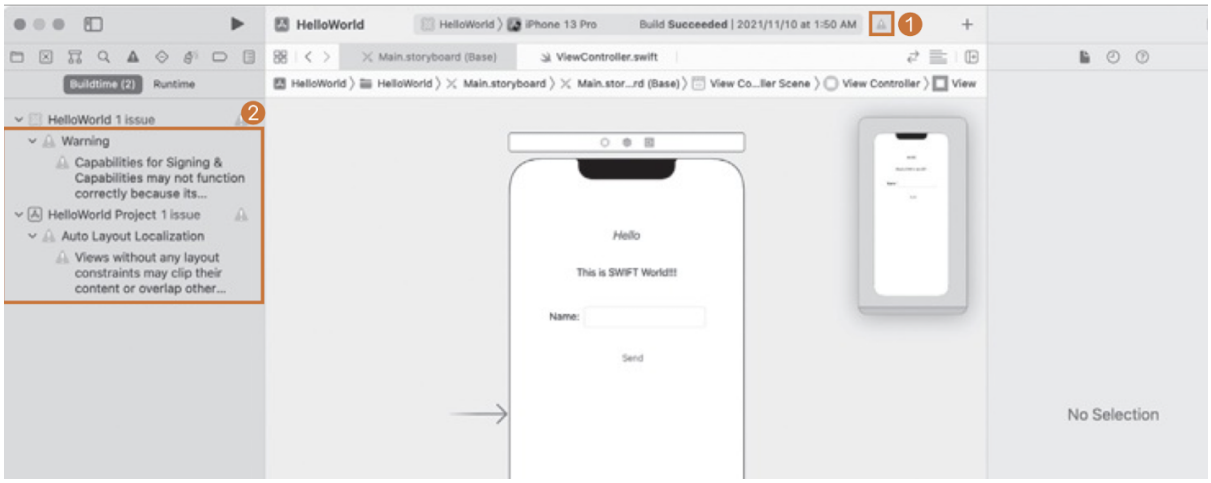


02-3 스토리보드로 Hello World 앱 화면 꾸미기

이제 스토리보드를 사용하여 Hello World 앱의 화면을 꾸며 보겠습니다. 이 앱에서는 텍스트를 보여주는 레이블(Label) 객체와 사용자가 직접 글자를 입력할 수 있는 텍스트 필드(Text Field) 객체, 이름을 전송하는 버튼(Button) 객체를 사용합니다.
객체란 사용자 인터페이스를 위해 사용하는 레이블, 버튼 등의 오브젝트를 의미하며 라이브러리에서 가져와 사용할 수 있습니다. 스토리보드에 배치된 객체는 사용자로부터 입력을 받거나 사용자 인터페이스 역할을 합니다.

위 그림은 완성된 스토리보드 화면입니다. 이 그림과 사용된 객체를 참고하여 배치해 보겠습니다.

경고 메시지 알아보기



- 예제를 작성하다 보면 경고 아이콘이 나타나는 것을 볼 수 있다.
- 이 경고 메시지는 자동 레이아웃에 관련된 메시지로 스토리보드에 객체를 배치시키면 나온다.

02-4 아웃렛 변수와 액션 함수 추가하기

아웃렛 변수와 액션 함수란?

- 스토리보드에 추가한 객체를 선택하고 내용을 변경하거나 특정 동작을 수행하도록 하기 위해 필요한 것
- `아웃렛 변수`
  - 해당 객체에 접근할 수 있는 변수
- `액션 함수`
  - 동작을 정의한 함수

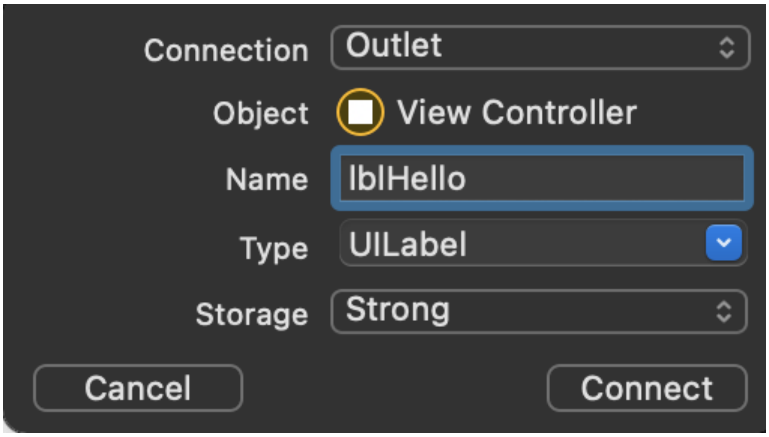
소스 작업을 위한 보조 편집기 영역 열기

레이블, 텍스트 필드에 아웃렛 변수 추가하기

1. 레이블에 아웃렛 변수 추가하기

- 컴포넌트를 마우스 오른쪽 버튼으로 선택한 후 오른쪽 보조 편집기 영역으로 드래그
- 아웃렛 변수는 일반적으로 클래스 선언부 바로 아래에 추가한다

2. 아래 연결 설정 창에서 연결(connection)이 [Outlet]으로 되어 있는 것을 확인한 후 아웃렛 변수의 이름 입력란에 ‘lblHello’라고 입력하고 타입(Type)이 ’UILabel’인지 확인



 상수와 변수

- 상수
  - `let`을 사용해 선언
  - 값을 지정해 주어야 하며 값이 한 번 결정되면 이후에는 값을 바꿀 수 없다
  - ex) let pi = 3.141592
- 변수
  - `var`를 사용해 선언
  - 최초 선언한 값 이외에도 중간에 계속해서 다른 값으로 변경 가능
  - var score = 95

 이름을 지정할 때 지켜야 하는 규칙

- 상수와 변수, 함수, 클래스의 이름을 지정할 때는 다음과 같은 규칙을 지켜야 한다.

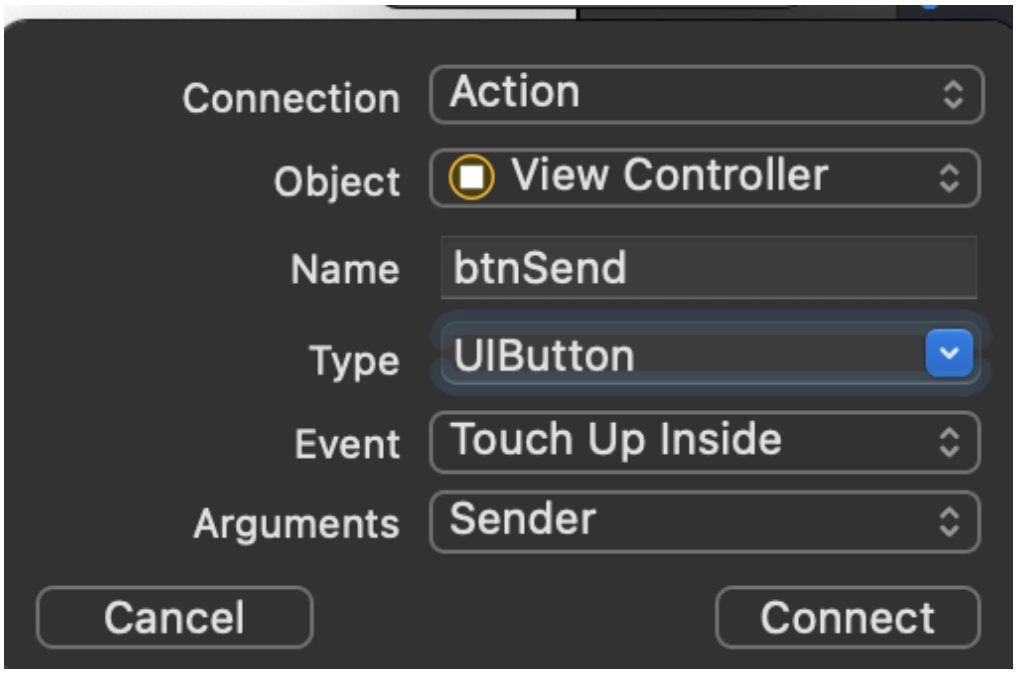
1. 유니코드 포함한 어떤 문자든 사용 가능
   단, 특수문자, 수학 기호, 화살표, 선, 상자, 그리기용 문자 사용 불가
2. 숫자로 시작할 수 없다.
3. 클래스 이름은 대문자의 명사로 시작
4. 함수나 메서드 이름은 소문자의 동사로 시작
5. 변수나 상수의 이름은 소문자의 명사로 시작
6. 시작 단어를 제외한 모든 단어의 시작은 대문자로 하고 그 외의 모든 문자는 소문자로

3. 텍스트 필드에 아웃렛 변수 추가하기

- 이름 입력란에 'txtName'

버튼에 대한 액션 함수 추가하기

1. [Send] 버튼을 클릭한 후 보조 편집기 영역으로 드래그



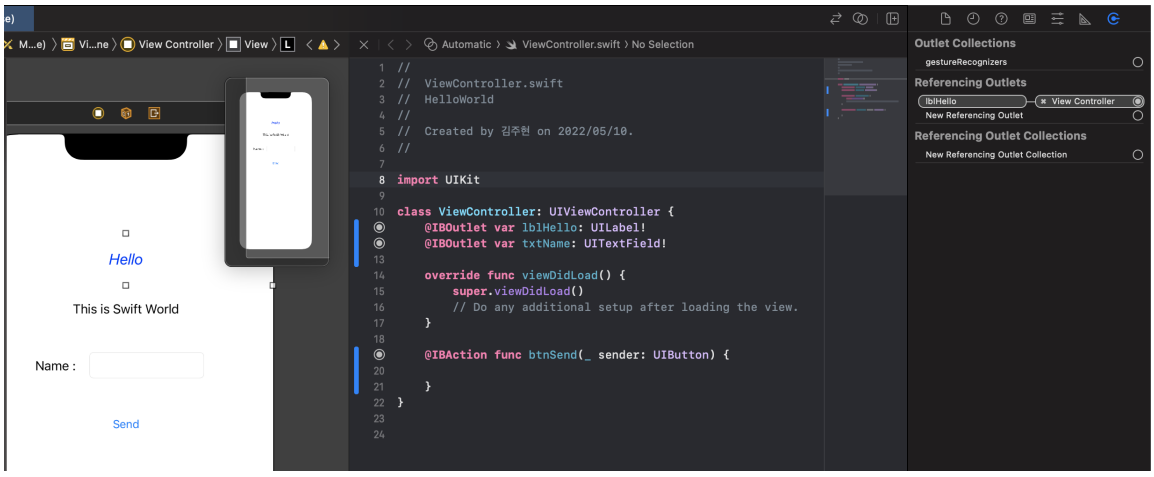
- 연결이 [Action]으로 되어 있는 것을 확인

> 💡 아웃렛 변수/액션 함수를 추가/삭제할 때 주의사항

- 아웃렛 변수 및 액션 함수를 추가하면 코드 왼쪽에 원으로 표시가 되는데, 이것은 스토리보드의 컴포넌트와 소스 코드가 연결되었음을 의미한다.

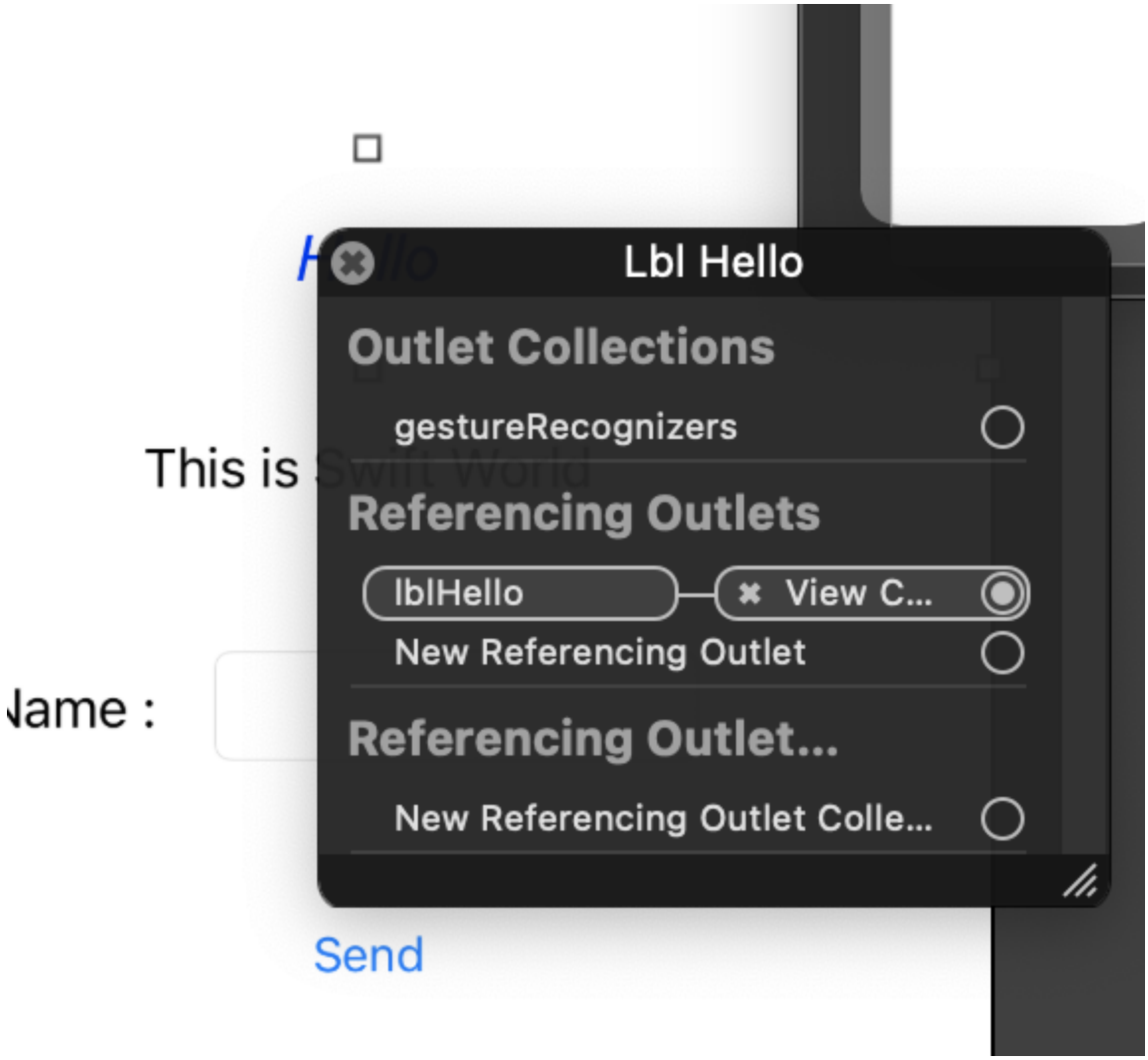
- 객체를 선택한 후 인스펙터 영역의

  [Show the Connection inspector] 버튼을 클릭하면 확인 가능



- 객체를 선택한 후 마우스 오른쪽 버튼을 눌러도 확인 가능

- 이 연결을 끊으려면 [X]를 클릭해 연결 상태를 삭제해야 한다.



02-5 버튼 클릭 시 동작 구현하기

소스 파일 열고 코딩하기

1. 버튼 클릭 시 동작할 함수 코딩하기

💡 문자열 합치기

- 스위프트에서는 문자열을 합칠 때 '+' 연산자 사용

```
lblHello.text = "Hello, " + "JooHyun"
```

💡 스위프트는 문장 맨 끝에 ';' 가 없음

시뮬레이터 실행하기

파일 저장하기

- 프로젝트를 완성하고 나서 [실행]버튼을 클릭하면
  Xcode는 모든 프로젝트를 자동으로 저장한 후 실행한다.
- 그렇지 않고 직접 프로젝트를 저장하고 싶다면 대부분의
  macOS 앱에서 저장하듯이 [command + S]

아웃렛 변수와 액션 함수에서 사용된 문법 뜯어보기

아웃렛 변수를 추가하는 소스

```
...
@IBOutlet var lblHello: UILabel!
    ①        ②         ③
```

위 소스는 02장에서 만든 Hello World 앱의 소스 중 일부를 발췌한 것입니다.

1. `@IBOutlet`

- @IBOutlet으로 정의된 변수를 아웃렛 변수라 부른다.
- @IB로 시작되는 변수나 함수는 인터페이스 빌더와 관련된 변수나 함수라는 것을 의미
- @IBOutlet 객체를 소스 코드에서 참조하기 위해 사용하는 키워드이며 주로 색상, 크기, 모양, 선의 두께, 텍스트 내용 등 객체의 속성을 제어하는 데 사용

2. `var lblHello:`

- 변수를 선언할 때는 var 키워드를 사용

3. `UILabel!`

- 선언하고자 하는 변수의 타입을 나타낸다.
- 위 예제에서는 레이블 객체에 대한 변수를 선언하는 것이므로 UILabel 클래스 타입을 선택

4. `strong / weak`

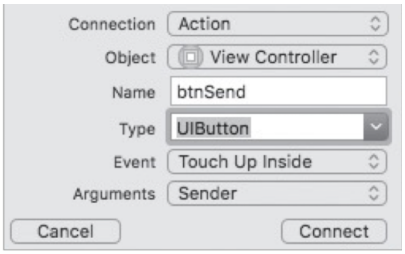
- 메모리 회수 정책을 나타내는 키워드
- 일반적으로 객체를 참조하기 위한 아웃렛 변수는 `strong`을 사용
- Xcode에서도 strong이 기본값으로 선택.
  weak로 선언하면 변수 정의 앞에 weak가 추가된다.

```
@IBOutlet var lblHello: UILabel!          // strong으로 선언
@IBOutlet weak var lblHello: UILabel!     // weak로 선언
```

액션 함수를 추가하는 소스 분석하기

```
...
@IBAction func btnSend(_ sender: UIButton) {
    ❶          ❷              ❸

    lblHello.text = "Hello, " + txtName.text!
        ❹       ❺      ❻     ❼       ❽
```



1. `@IBAction:`

- 객체의 이벤트를 제어하기 위해 사용하는 키워드

2. `func btnSend`

- 함수를 선언할 때 func 키워드를 사용

3. `(_sender: UIButton)`

- 액션 함수가 실행되도록 이벤트를 보내는 객체
- 버튼 객체에서 이벤트가 발생했을 때 해당 액션 함수를 실행시킬 것이므로 UIButton 클래스 타입 선택