04 장 데이트 피커 사용해 날짜 선택하기

데이트 피커(Date Picker)는 아이폰에서 날짜를 선택할 때 사용하는 객체이다.
아이폰을 사용한다면 데이트 피커 기능 역시 한 번쯤은 사용해 보았을 것이다.
우리가 흔히 알람을 설정하거나 일정을 기록할 때 날짜를 설정하는 화면에서 빠지지 않는 기능이다.

04-1 데이트 피커란?

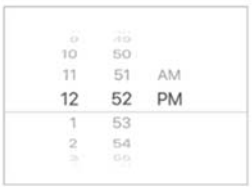
데이트 피커(Date Picker)는 날짜와 시간을 선택할 수 있게 해주는 객체로서, 아이폰의 기본 기능인 시계 앱의 알람 탭에서 자주 사용하는 기능 중 하나이다.

시계 앱

미리 알림 앱

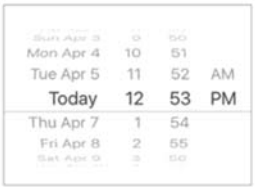
데이원(Dayone) 앱

데이트 피커는 시간형, 날짜형, 날짜 & 시간형, 카운트다운형, 이렇게 네가지 모드를 제공하며 각 모드는 화면에 다음과 같이 표시된다.



시간형(Time)

날짜형(Date)



날짜&시간형(Date and Time)

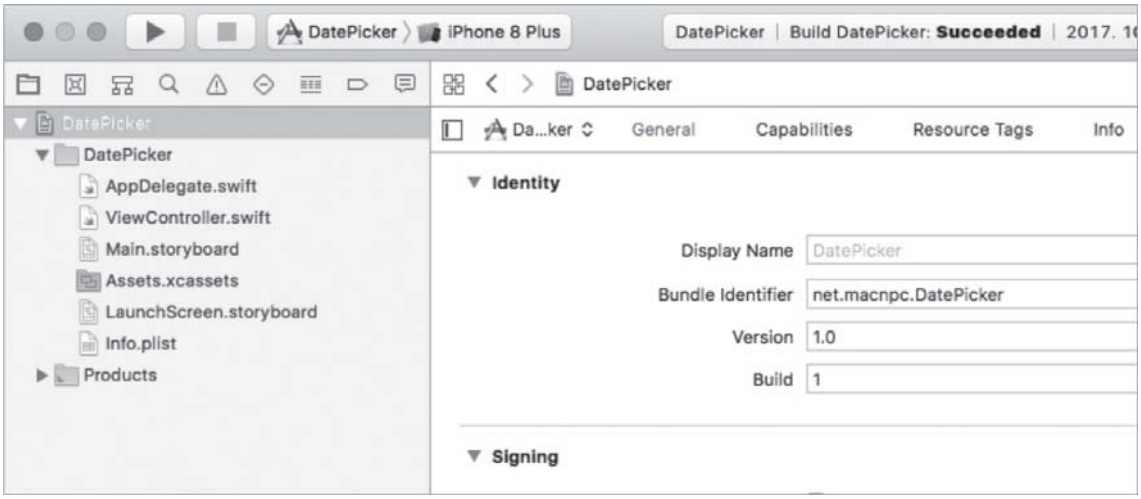
카운트다운형(Countdown)

04-2 데이트 피커 앱을 위한기본 환경 구성하기

이 앱에서는 데이트 피커를 사용해 날짜를 선택하고 선택한 날짜를 화면에 출력해 주는 기능을 구현해 보자.

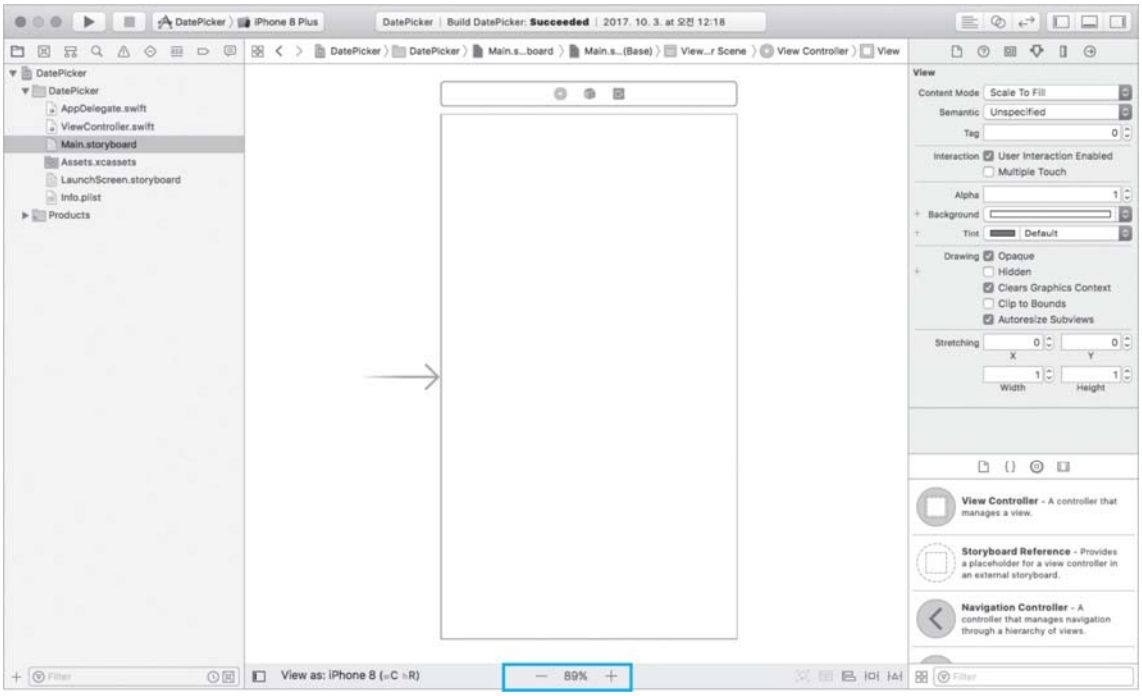
또한 타이머를 사용해 정해진 시간마다 주기적으로 현재 시간을 출력하는 방법을 알아보자.

1. 먼저 'DatePicker'라는 이름으로 새 프로젝트를 만든다.



2. 뷰 컨트롤러 크기 조절하기

아이폰 모양의 뷰 컨트롤러가 화면에 꽉 차있다. 모니터가 작아 답답하다면 아랫부분의 [Zoom In], [Zoom Out] 버튼을 사용하여 뷰 컨트롤러의 크기를 조절할 수 있다,

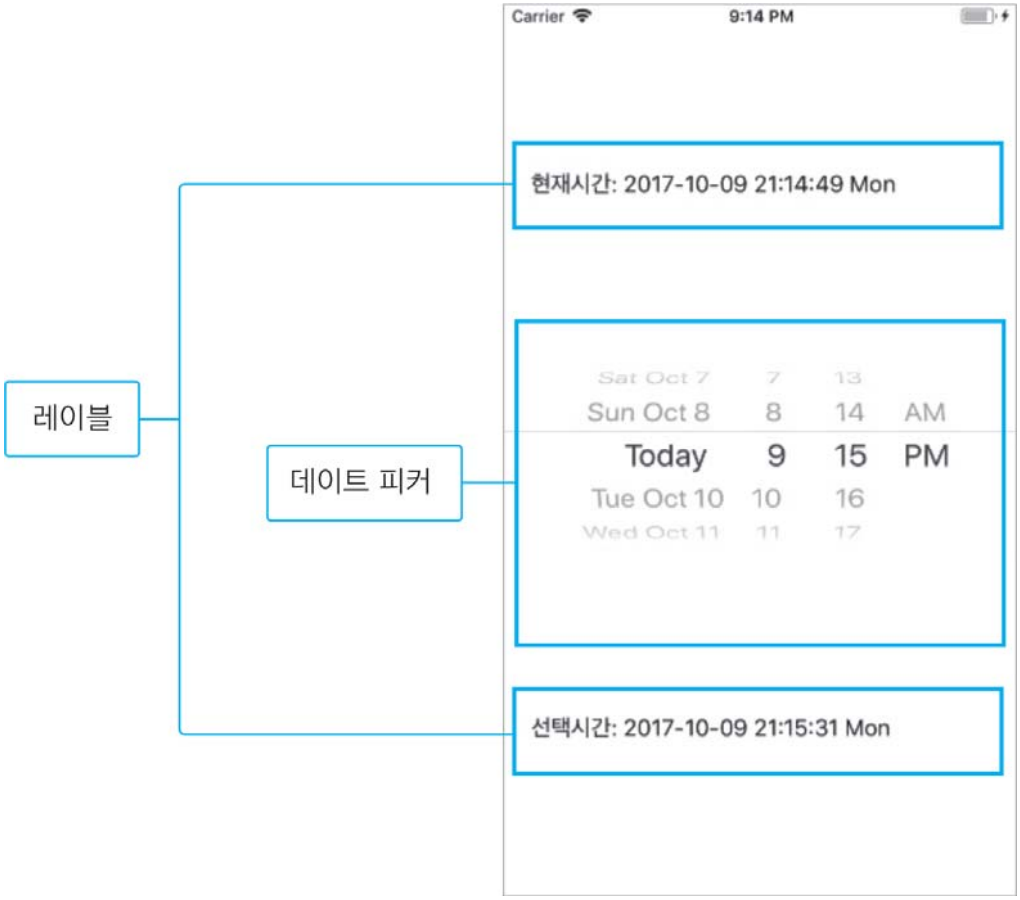


04-3 스토리보드로 데이트 피커 앱 화면 꾸미기

완성된 스토리보드 화면입니다.

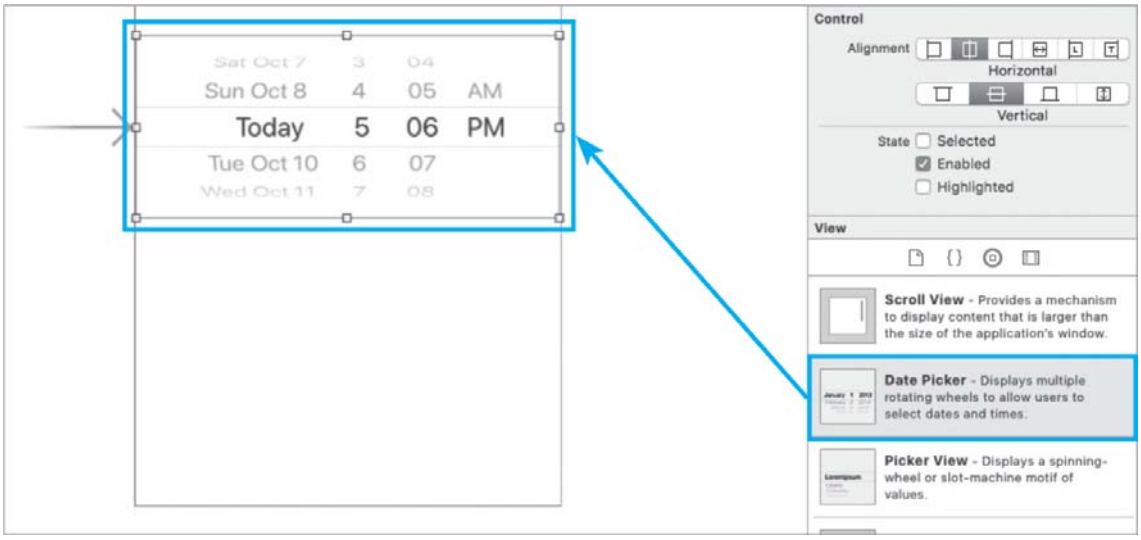
이 그림과 사용된 객체를 참고하여 배치해 보겠습니다.

사용된 객체: 데이트 피커, 레이블

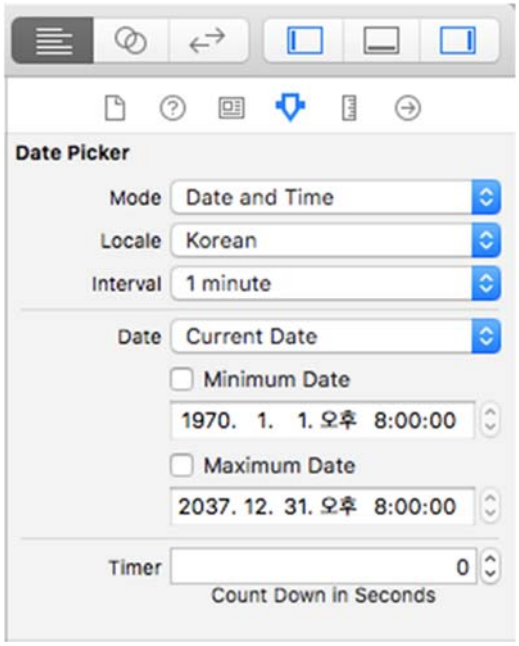


1. 데이터 피커 추가하기

오른쪽 아랫부분의 오브젝트 라이브러리를 스크롤하여 [데이트 피커(Date Picker)]를 선택한 후 데이트 피커를 그대로 스토리보드로 끌어와 화면의 중간에 배치한다.



만일 데이트 피커의 모드를 변경하고 싶으면 데이트 피커 객체를 선택하고 오른쪽 [Attributes inspector]버튼을 클릭한 후 Mode 를 변경하면 된다. 여기서는 기본 설정인 [Date and Time]을 사용하겠다.



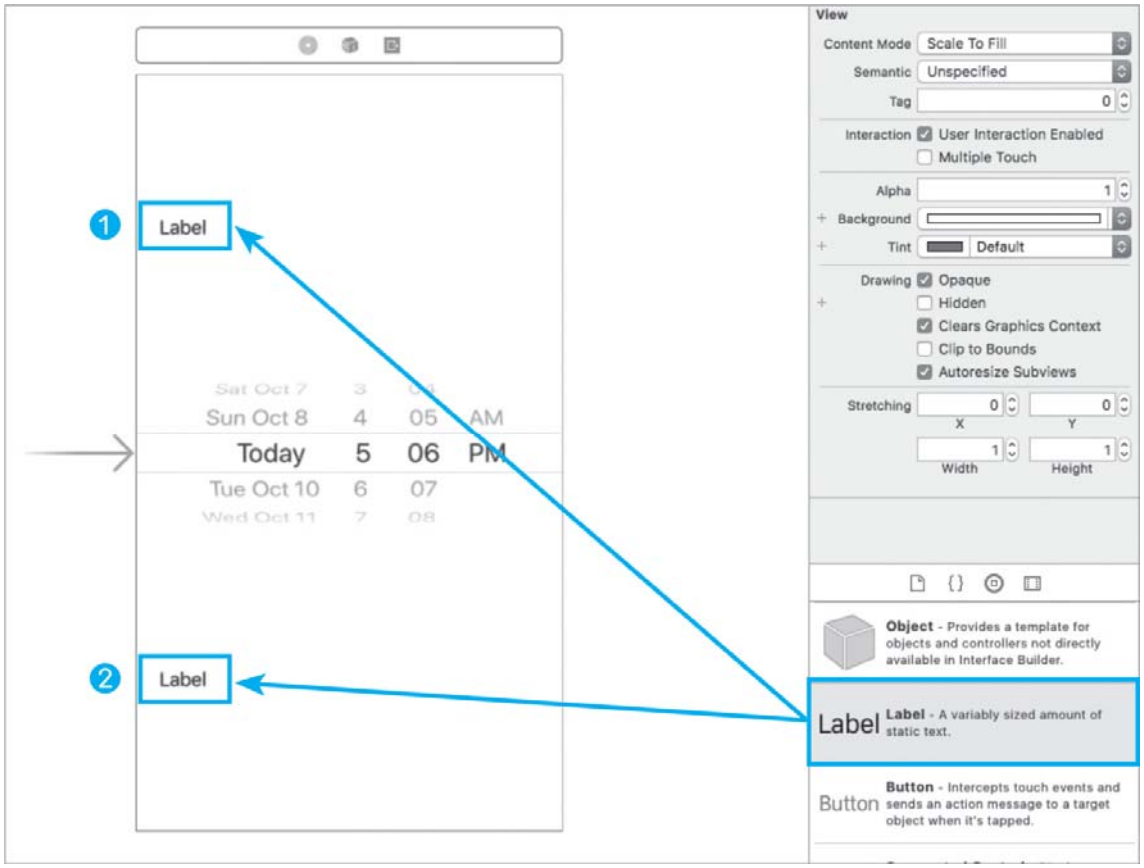
데이트 피커의 날짜 표기를 한국어로 바꾸려면 [Atrribute inspector]에서 Location 을 [Korean]으로 변경하면 된다.

2. 레이블 추가하기

먼저 현재 시간을 보여주기 위한 레이블을 추가하자.

오브젝트 라이브러리에서 [레이블(Label)]을 선택한 후 스토리보드로 끌어와 데이트 피커 왼쪽 윗부분에 배치한다.

화면 아래쪽에는 데이트 피커를 사용해서 선택한 시간을 보여 줄 레이블을 추가한다.

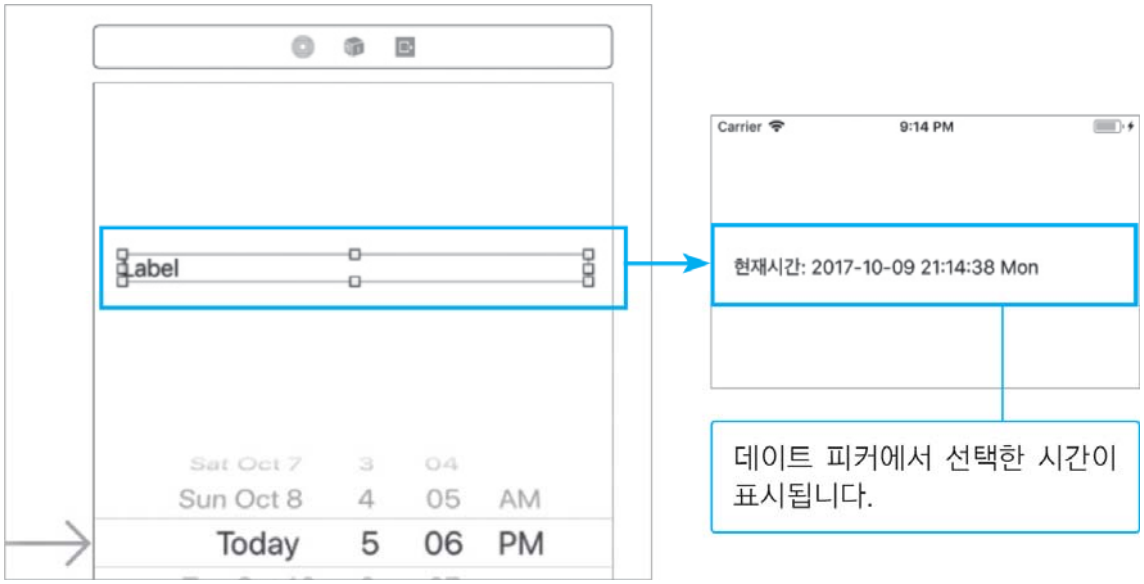


3. 스토리보드 수정하기

스토리보드를 약간 수정해 보자. 먼저 위쪽 레이블의 크기를 화면 가로 폭에 맞게 넓힌다.

위쪽 레이블을 마우스로 더블 클릭하여 텍스트를 '현재시간 : '으로 수정한다,

두번째 레이블도 더블 클릭하여 텍스트를 '선택시간 : '으로 수정한다.



04-4 아웃렛 변수와 액션 함수 추가하기

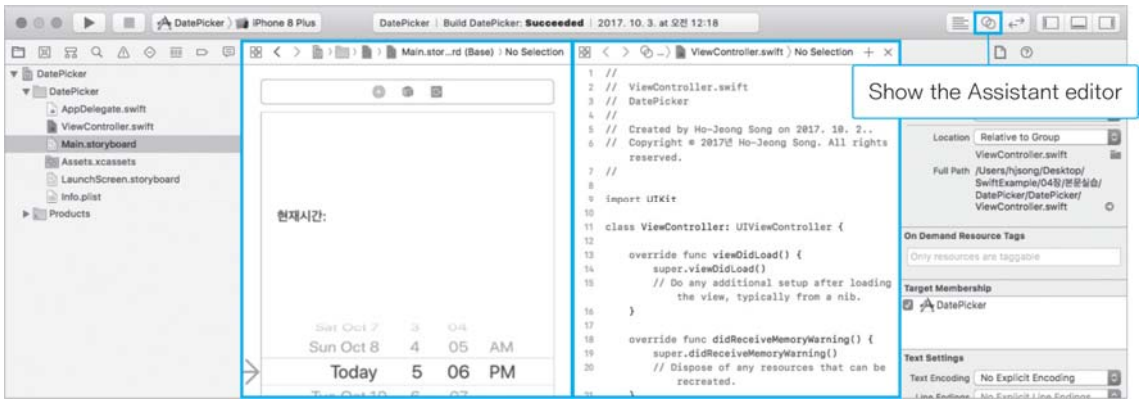
아웃렛 변수와 액션 함수를 추가한다.

데이트 피커를 사용해 날짜를 선택했을 때 선택한 날짜를 표시하기 위한 데이트 피커의 아웃렛 변수와 선택한 날짜를 레이블에 출력하기 위한 레이블의 아웃렛 변수 그리고 현재 시간을 레이블에 출력하기 위한 레이블의 아웃렛 변수를 추가한다.

1. 보조 편집기 영역 열기

아웃렛 변수와 액션 함수를 추가하려면 우선 오른쪽 윗부분의 [Show the Assistant editor]버튼을 클릭하여 보조 편집기 영역(Assistant editor)을 열어야 합니다.

가운데 화면의 스토리보드 부분이 둘로 나눠지면서 왼쪽에는 스토리보드, 오른쪽에는 소스를 편집하는 영역이 나타난다.

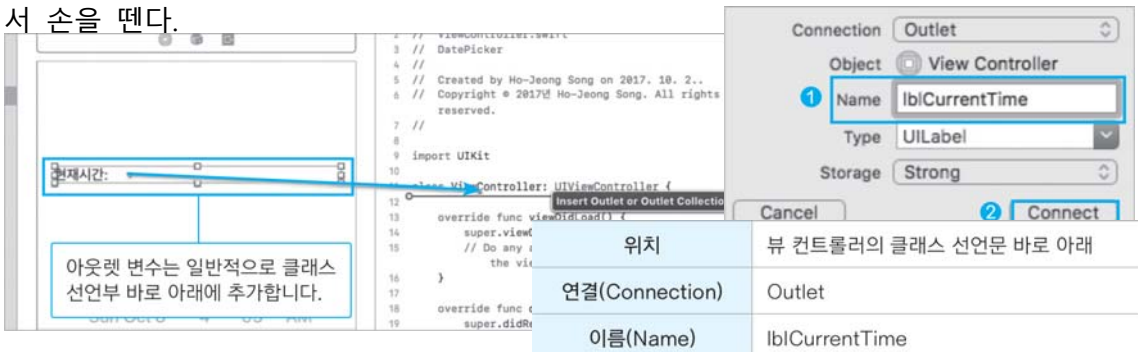


2. 레이블에 대한 아웃렛 변수 추가하기

우선 첫번째로 '현재시간 : '이라고 입력한, 위쪽 레이블에 대한 아웃렛 변수를 추가한다. 레이블에 대한 아웃렛 변수를 사용하여 추후 코딩 작업을 할 때 레이블의 텍스트 값을 변경할 수 있다.

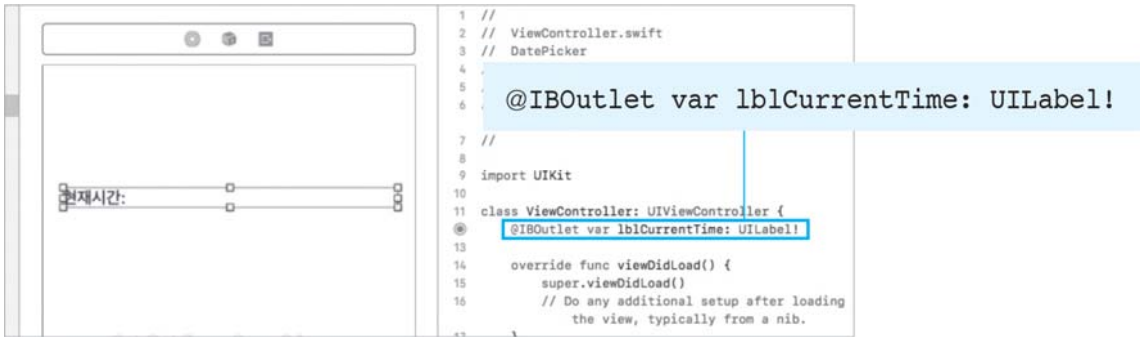
위쪽의 레이블을 마우스 오른쪽 버튼으로 선택한 후 오른쪽 보조 편집기 영역으로 드래그하면 아래 그림과 같이 연결선이 나타난다.

이 연결선을 뷰 컨트롤러의 클래스(class)선언문 바로 아래에 배치한 후 마우스 버튼에서 손을 뗀다.

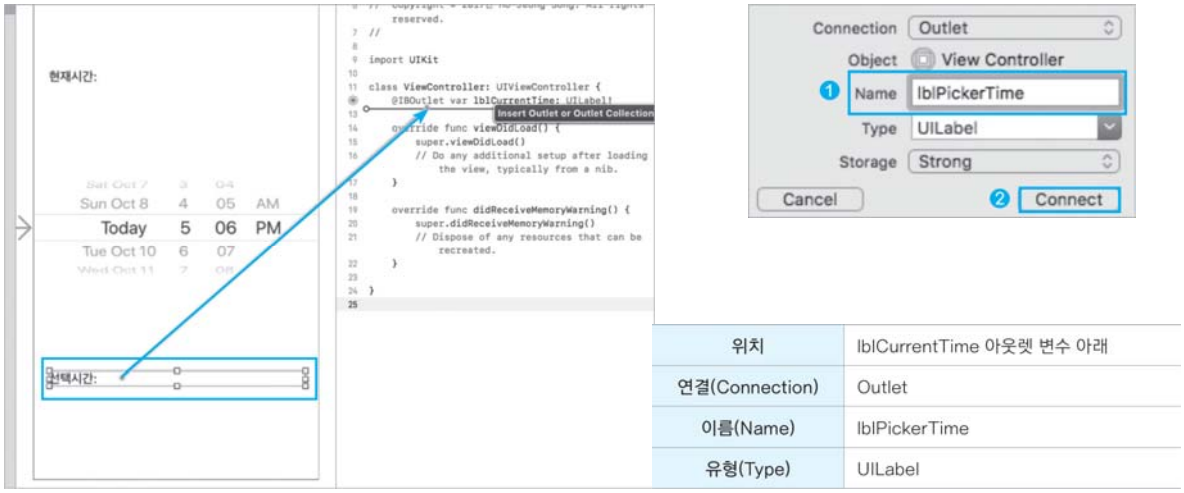


위치	뷰 컨트롤러의 클래스 선언문 바로 아래
연결(Connection)	Outlet
이름(Name)	lblCurrentTime
유형(Type)	UILabel

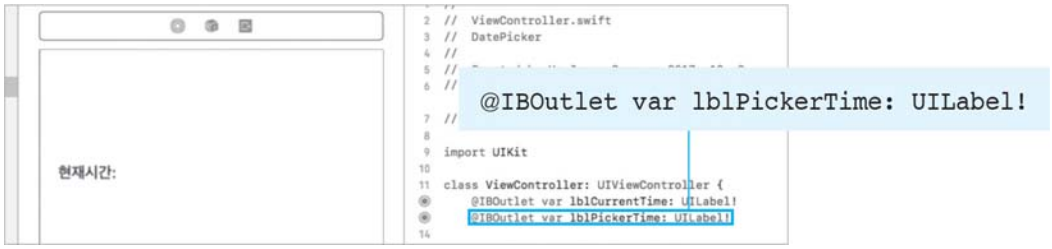
그리고 설정 창에서 아웃헷 변수의 이름(Name)을 'lblCurrentTime'으로 입력한 후 [Connect]버튼을 클릭하여 레이블과 아웃렛 변수를 연결한다.



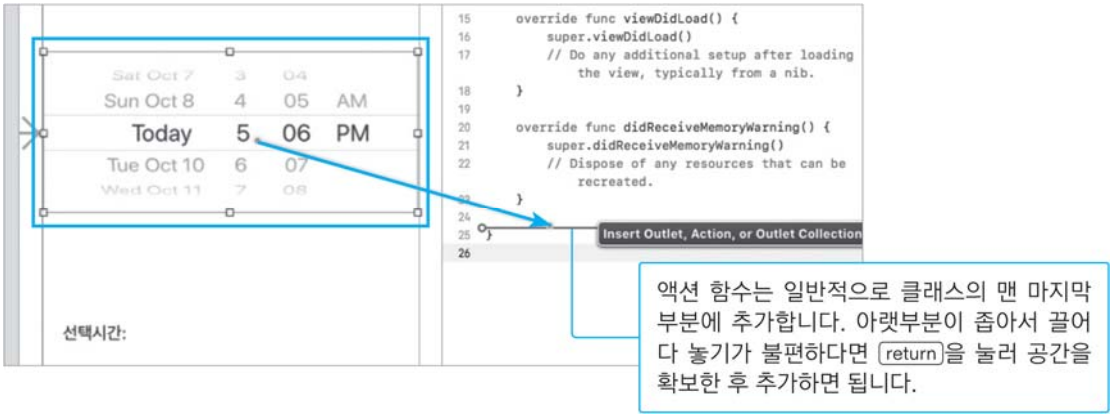
3. 두번째로 '선택시간 : '이라고 입력한, 아래쪽 레이블에 대한 아웃렛 변수를 추가한다. 위쪽 레이블에 대한 아웃렛 변수를 추가하는 방법과 같다.
[선탠시간 :]레이블을 마우스 오른쪽 버튼으로 선택한 후 드래그해서 오른쪽 보조 편집기 영역의 조금전에 추가한 'lblCurrentTime' 변수 아래로 끌어다 놓는다.



위치	lblCurrentTime 아웃렛 변수 아래
연결(Connection)	Outlet
이름(Name)	lblPickerTime
유형(Type)	UILabel

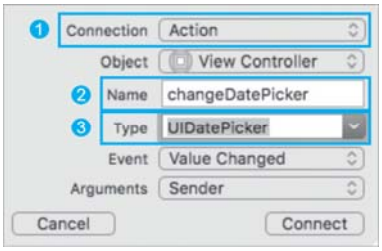


4. 데이트 피커에 대한 액션 함수 추가하기
데이트 피커에 대한 액션 함수를 추가한다. 데이트 피커에 대한 함수는 데이트 피커를 선택했을 때 실행된다.
레이블에 대한 아웃렛 변수 추가하기와 방법은 같다. 마우스 오른쪽 버튼으로 [Date Picker]를 선택한 후 드래그해서 오른쪽 보조 편집기영역의 아랫부분에 끌어다 놓는다.

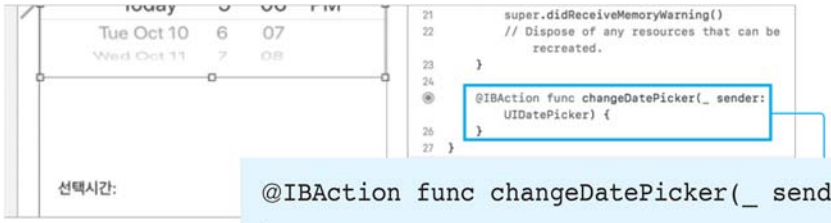


5. 설정 창이 나타나면 연결(Connection)을 [Action]으로 변경한다.
그리고 이름(Name)을 'changeDatePicker'로 입력한 후 타입(Type)을 해당 객체에 맞는 [UIDate Picker]로 변경한다.

위치	Class ViewController 닫음 괄호'}' 바로 위
연결(Connection)	Action
이름(Name)	changeDatePicker
유형(Type)	UIDatePicker



6. 데이트 피커에 대한 액션 함수가 추가되었다.




```
@IBAction func changeDatePicker(_ sender: UIDatePicker) {
}
```

이렇게 데이트 피커와 레이블 아웃 변수를 추가하였다.
데이트 피커 아웃렛 변수를 이용하여 레이블에 접근할 수 있게 된다.

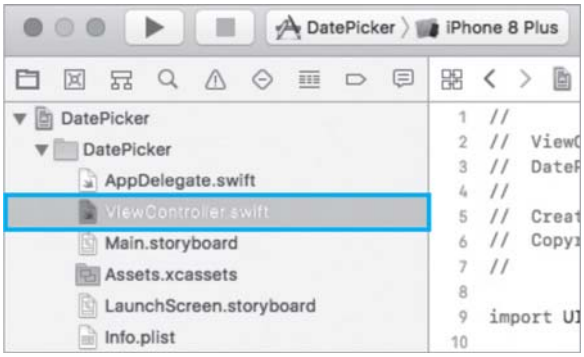
04-5 선택한 날짜와 시간을 출력할 코드 작성하기

데이터 피커의 롤렛을 돌렸을 때 선택한 날짜 및 시간을 출력하는 방법에 대해서 알아보자.

데이트 피커로 선택한 날짜와 시간을 출력하기

1. 오른쪽 윗부분에서 [Show the Standard editor]버튼을 클릭한다. 그러면 스토리보드와 보조 편집기 영역으로 나누어져 있던 화면이 하나의 화면으로 바뀌게 된다.

2. 왼쪽의 내비게이터 영역에서 [ViewController.swift]를 선택한다.



3. 선택한 날짜 출력하기

사용자가 데이트 피커에서 원하는 날짜와 시간을 선택하면 그 내용이 'lblPickerTime'레이블에 출력되는 기능을 구현해 보자.

changeDatePicker 에 다음 코드를 추가한다.

```
21          super.didReceiveMemoryWarning()
22          // Dispose of any resources that can be recreated.
23      }
24
25      @IBAction func changeDatePicker(_ sender: UIDatePicker) {
26          let datePickerView = sender
27
28          let formatter = DateFormatter()
29          formatter.dateFormat = "yyyy-MM-dd HH:mm:ss EEE"
30          lblPickerTime.text = "선택시간: " + formatter.string(from: datePickerView.date)
31      }
32  }
```

```
@IBAction func changeDatePicker(_ sender: UIDatePicker) {
      let datePickerView = sender ❶

      let formatter = DateFormatter() ❷
      formatter.dateFormat = "yyyy-MM-dd HH:mm:ss EEE" ❸
      lblPickerTime.text =
        "선택시간: " + formatter.string(from: datePickerView.date) ❹
}
```

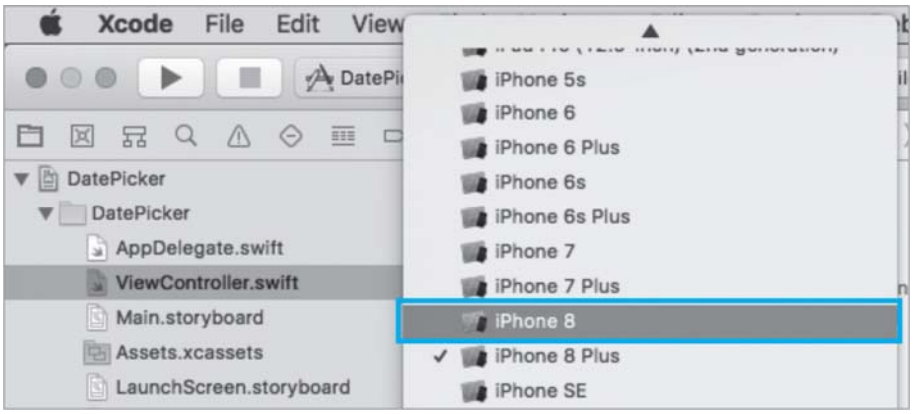
1) 데이트 피커를 선택할 때 발생하는 액션 함수인 'changDatePicker'가 호출되면서 sender 라는 UIDatePicker 자료형의 인수가 전달된다. 이 sender 를 datePickerView 라는 상수에 저장한다.

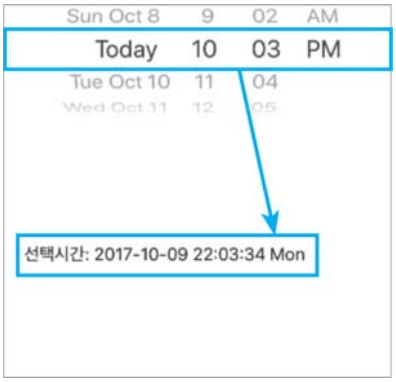
2) 날짜를 출력하기 위하여 DateFormatter 라는 클래스 상수 formatter 를 선언합니다.

3) formatter 의 dateFormat 속성을 설정한다. 포맷은 "년-월-일 시:분:초 요일"로 설정된다.

4) 데이트 피커에서 선택한 날짜를 formatter 의 dateFormat 에서 설정한 포멧대로 String 메서드를 사용하여 문자열(String)로 변환한다. "선택시간: "이라는 문자열에 위에서 문자열로 변환한 date 값을 추가하여 lblPickerTime text 에 넣는다.

4. 결과 보기





[스위프트 문법] 데이터 포맷의 형식

필드	심벌	결과	의미
년(Year)	yy	16	두 자리로 연도를 표시
	yyyy	2016	네 자리로 연도를 표시
월(Month)	M	5	한 자리로 월을 표시
	MM	05	두 자리로 월을 표시
	MMM	Mar	Jan ~ Dec까지 3글자만 영문으로 월을 표시
	MMMM	March	January ~ December까지 전체를 영문으로 월을 표시
주(Week)	w	6	1~52까지 연간 주 순서(week of year)를 표시
	ww	13	01~52까지 연간 주 순서(week of year)를 표시
	W	5	1~6까지 월간 주 순서(week of month)를 표시
일(Day)	d	8	1~31까지 일을 표시
	dd	08	01~31까지 일을 표시
	D	35	1~366까지 연간 일 순서(day of year)를 표시
	DD	35	01~366까지 연간 일 순서(day of year)를 표시
	DDD	035	001~366까지 연간 일 순서(day of year)를 표시
요일 (Weekday)	E, EE, EEE	Mon	Sun ~ Sat까지 3글자 요일 표시
	EEEE	Monday	Sunday ~ Saturday까지 요일 전체 이름 표시
	EEEEE	M	1글자 약어 요일 표시
	e	4	1~7까지 주간 날짜 순서 표시
	ee	04	01~07까지 주간 날짜 순서 표시
시기(Period)	a	PM	AM/PM 표시

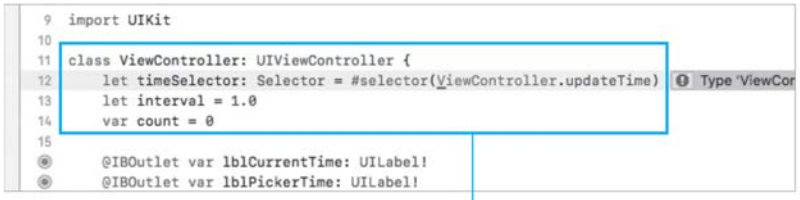
시간(Hour)	h	3	1~12까지 시각을 표시
	hh	03	01~12까지 시각을 표시
	H	15	1~24까지 24시간 시각을 표시
	HH	15	01~24까지 24시간 시각을 표시
분(Minute)	m	36	0~59까지 분을 표시
	mm	36	00~59까지 분을 표시
초(Second)	s	44	0~59까지 초를 표시
	ss	44	00~59까지 초를 표시
지역(Zone)	z	GMT+09:00	타임존 표시
	Z	+0900	GMT 시간차 표시

04-6 타이머 기능 추가하기

이제는 iOS 에서 타이머를 사용하는 방법에 대하여 알아본다.
타이머는 정해진 시간에 한 번씩 설정한 함수를 실행하는 기능이다.
이 타이머 기능을 사용하여 1 초에 한 번씩 현재 시간을 레이블에 출력해 보자.

1. 변수 및 상수 추가하기

우선 타이머응 구동하는 데 필요한 변수 및 상수를 추가해 보자.
다음 소스를 classViewController: UIViewController { 아래에 추가하자.

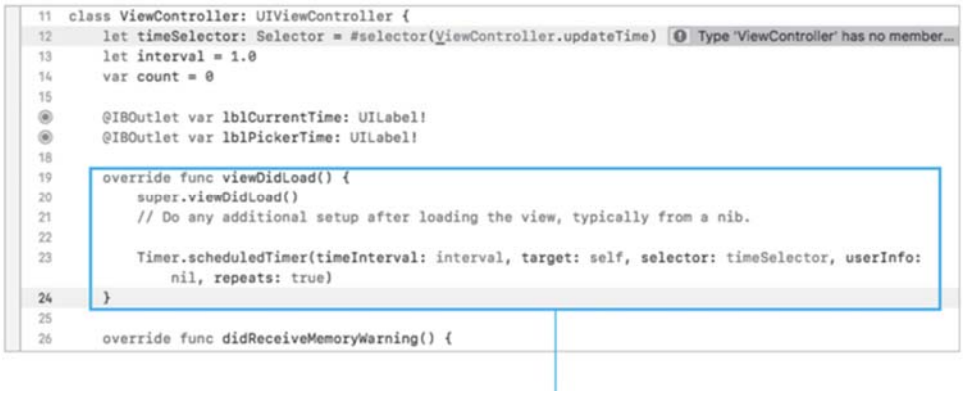



```
class ViewController: UIViewController {
    let timeSelector: Selector = #selector(ViewController.updateTime) ❶
    let interval = 1.0 ❷
    var count = 0 ❸
      ⋮
}
```

1) timeSelect : 타이머가 구동되면 실행할 함수를 지정한다.
updateTime()이라는 함수가 없어서 오류가 발생할 수 있다. 여기서는 무시하고 넘어가자.
2) interval : 타이머의 간격 값이다. 1.0 은 1 초를 의미한다.
3) count : 타이머가 설정한 간격대로 실행되는지 확인하기 위한 변수이다.

2. 타이머 설정하기

타이머를 설정하기 위해 scheduledTime 함수를 사용한다.



```
override func viewDidLoad() {
        super.viewDidLoad()
       // Do any additional setup after loading the view, typically from a nib.
       Timer.scheduledTimer(timeInterval: interval, target: self,
              selector: timeSelector, userInfo: nil, repeats: true)
}
```

1) scheduledTime 의 각 인수는 다음과 같은 의미를 갖는다.

Wlp huvfkhgxdgWlp hu=wlp hIqwhuydo=#qwhuydo,#wdujhw=vhoi##

##########################타이머#간격###################동작될#뷰#

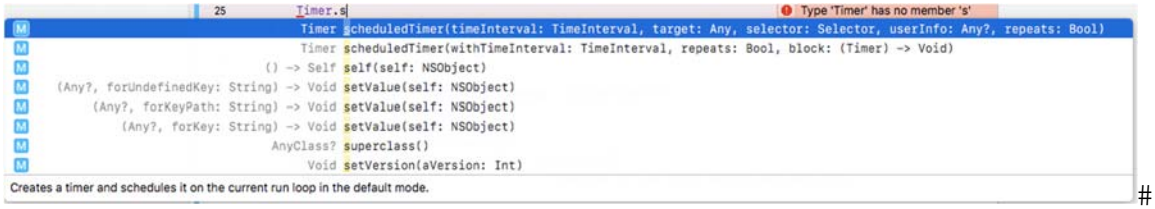
vhofwru##wlp hVhofwru/#xvhuIqir##qlo##hshdw=wrxh,#

################################사용자#정보#########반복여부#

#

소스를#입력할#때#Wlp huvf까지만#입력하면#[frgh에서#관련#메서드들을#자동으로#

찾아준다1##



#

자동으로#메서드가#입력되면#첫번째#인자가#선택되어#있고#인수를#입력한#후#^ds키를#

누르면#커서가#다음#인수#위치로#이동한다1#

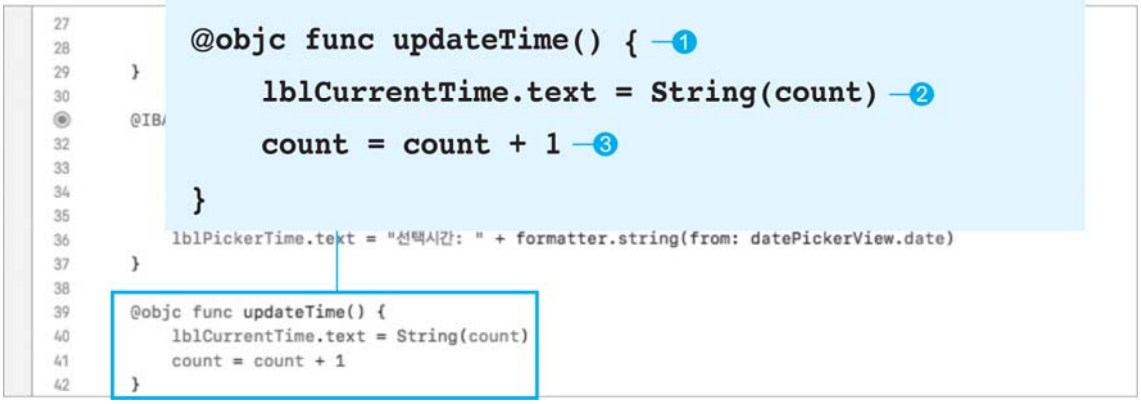
#

61#타이머#동작#함수#추가하기#

타이머가#동작할#때#실행할#함수를#추가하자1#

소스의#마지막#부분에#xsgdwhWlp h함수를#추가한다1#

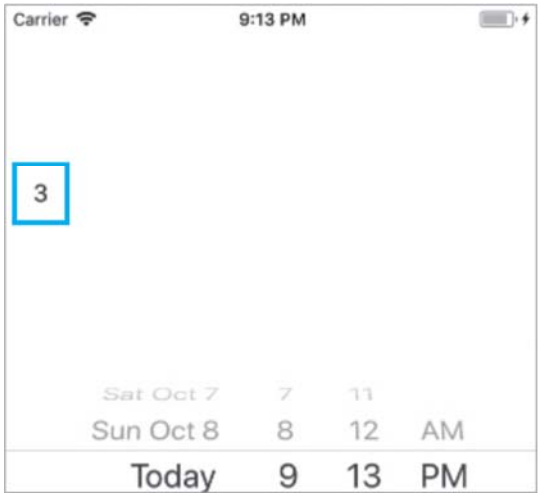
이#xsgdwhWlp h함수는#위의#wlp hVhofwru에서#정의한#이름과#같아야#한다1#

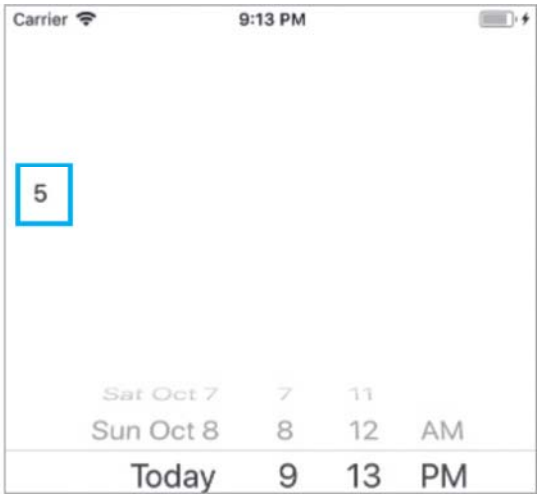


1) Swift4에서는 #selector()의 인자로 사용될 메서드를 선언할 때 Objective-C와 호환성을 위하여 함수 앞에 반드시 @objc키워드를 붙여야 한다.

2) String으로 변환한 count값을 'lblCurrentTime'레이블의 text속성에 저장한다.
3) count 값을 1 증가한다,

4. 결과 화면 확인하기
[실행]버튼을 클릭하여 결과를 확인한다. 'lblCurrentTime'레이블에 숫자가 0부터 시작해 1초에 1씩 증가하는 것을 확인할 수 있다.

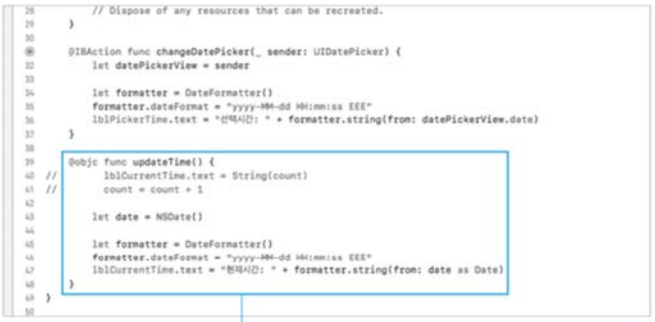




04-7 현재 시간을 읽는 함수 추가하기

현재 시간 출력하기

1. updateTime이라는 함수 안의 내용을 다음과 같이 수정한다.



```
    @objc func updateTime() {
//          lblCurrentTime.text = String(count)
//          count = count + 1

        let date = NSDate()

        let formatter = DateFormatter()
        formatter.dateFormat = "yyyy-MM-dd HH:mm:ss EEE"
        lblCurrentTime.text = "현재시간: " + formatter.string(from: date as Date)
    }
```

1) 기존 소스를 주석처리 했다.

2) 현재시간을 NSDate 함수를 사용하여 가져온다.

3) 날짜를 출력하기 위하여 DateFormatter 라는 클래스의 상수 formatter 를 선언한다.

4) 앞에서 선언한 상수 formatter 의 dateFormat 속성을 설정한다.

5) formatter.string(date)은 피커 뷰에서 선택한 날짜를 formatter 의 dateFormat 에서 설정한 포멧대로 string 메서드를 사용하여 문자열로 변환한다.

lblCurrentTime.text='현재시간 : " + 는 "현재시간 :"이라는 String 에 위에서 String 으로 변환한 date 값을 추가하여 lblCurrentTime 의 text 에 넣는다.

스위프트 문법] 주석 처리하기

다른 프로그래밍 언어와 마찬가지로 스위프트에서도 주석 처리를 할 수 있다.

코딩을 하다가 필요 없거나 잠시 사용하지 않는 문장 또는 문단이 있을 경우 주석으로 처리할 수 있다.

주석 처리는 다음과 같이 '//'와 '/*...*/' 기호를 사용한다.

1) '//'기호는 한 줄을 모두 주석 처리한다.

```
//    func pickerView(pickerView: UIPickerView, titleForRow row: Int,
          forComponent component: Int) -> String? {
//        return imageFileName[row]
//    }
```

2) '/*와 '*/'사이에 있는 문장을 주석처리한다.

```
/*
    func pickerView(pickerView: UIPickerView, titleForRow row: Int,
         forComponent component: Int) -> String? {
         return imageFileName[row]
    }
*/
```