Microsoft

Xamarin

한 발 앞서 배워보는
Xamarin

Android, iOS, Windows 앱 개발을 한번에

2016. 5. 27(금) 18:30 ~ 22:00

한국마이크로소프트 대회의실(11층)

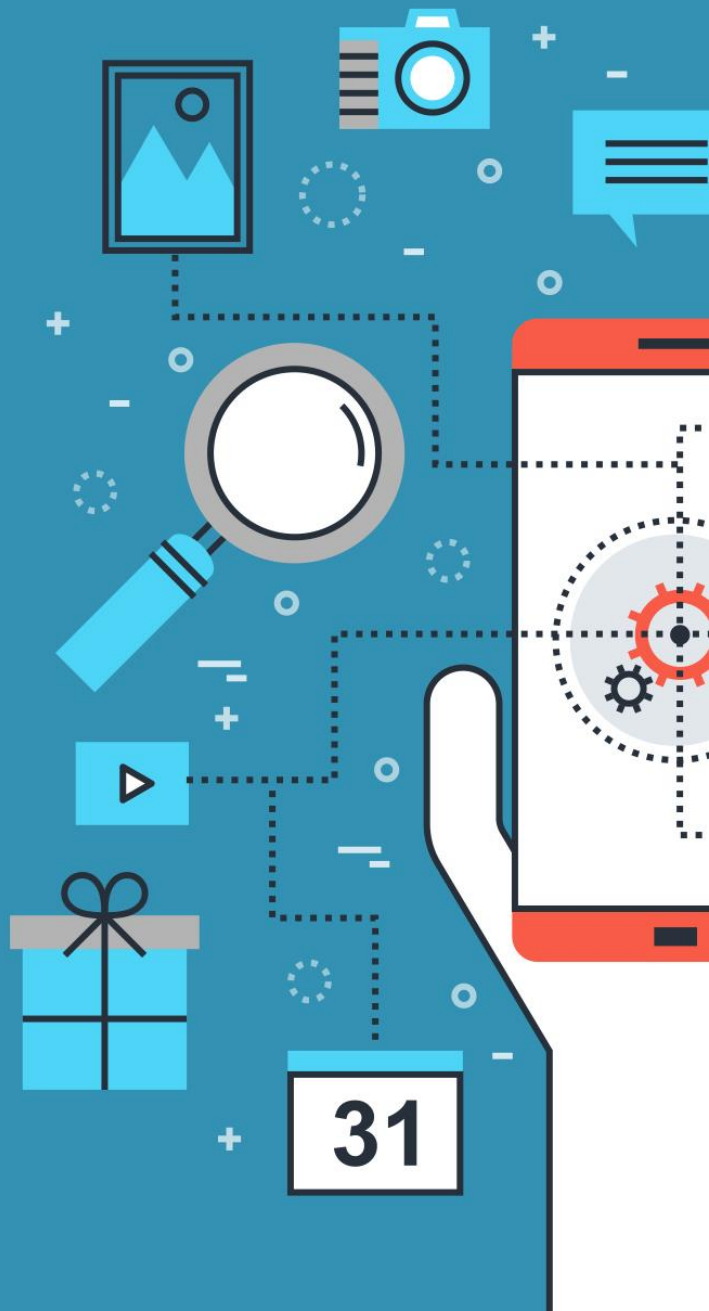
Xamarin 초보를 벗어나기 위한 필수 스킬

Xamarin Forms와 네이티브 계층 연동

이규원 / **envicase**

https://doc.co/N7VeYp

https://github.com/gyuwon/ms-xamarin-seminar-2016

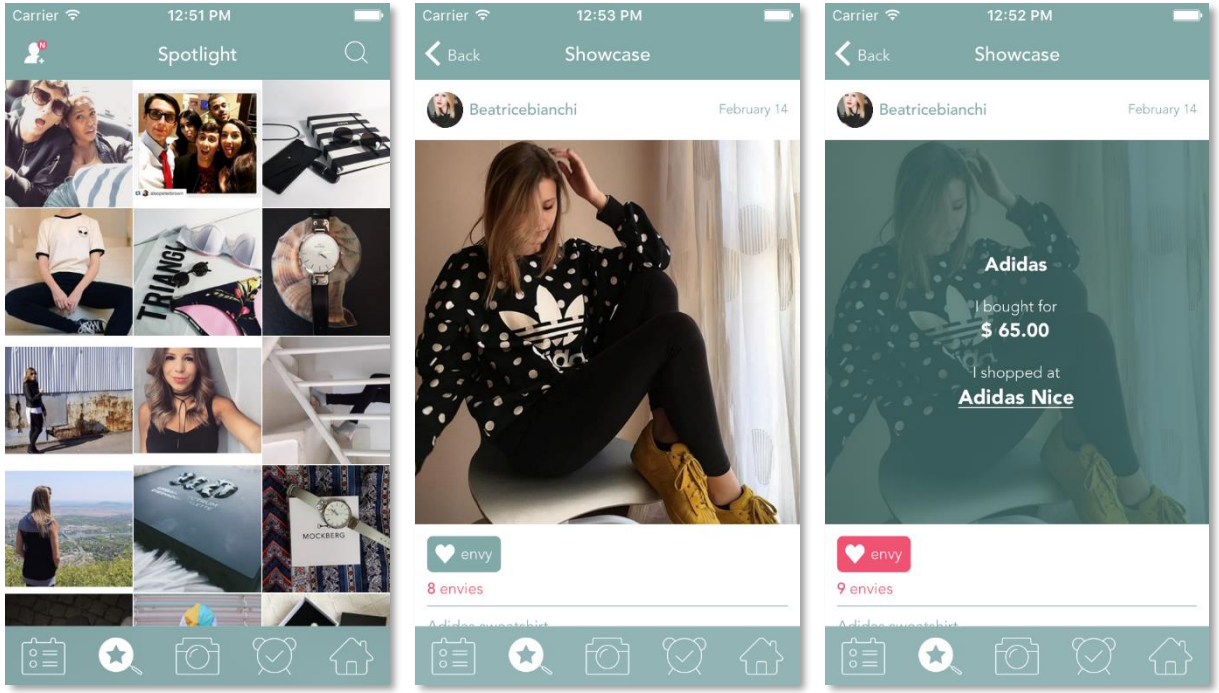


envicase

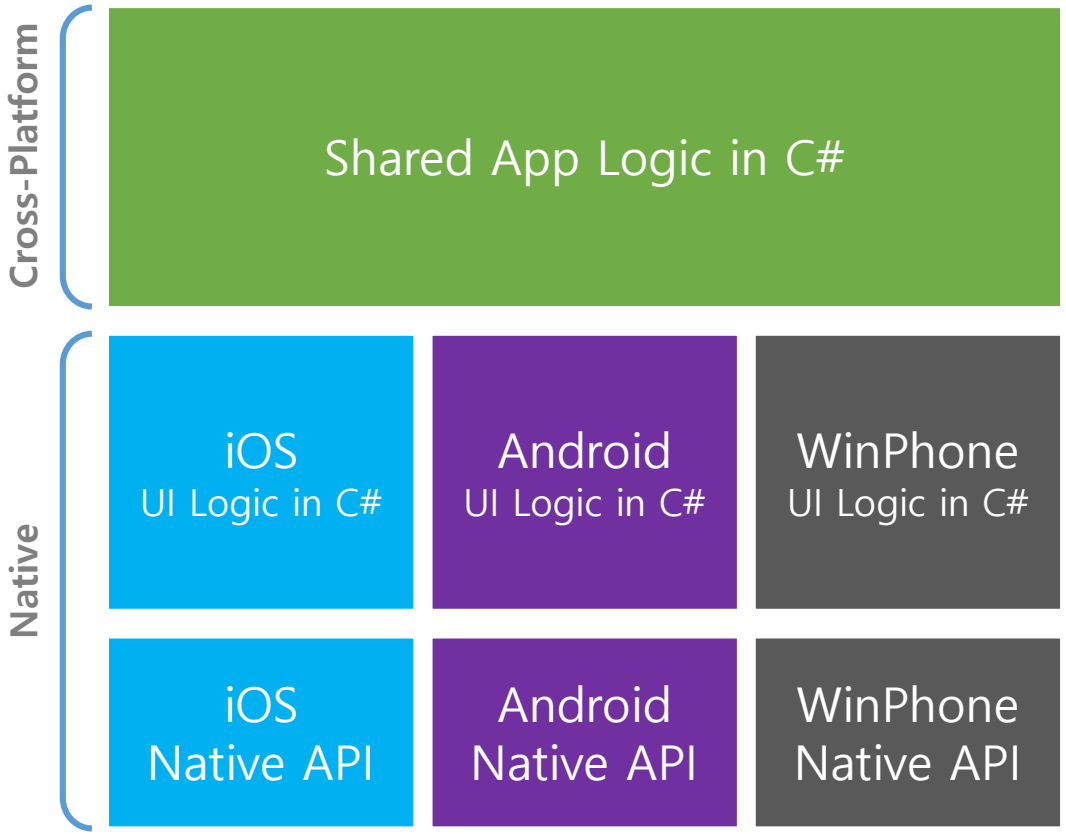
- 구매품과 애장품들을 전시하는 개인형 쇼룸 서비스
- Xamarin Forms로 iOS 클라이언트 개발

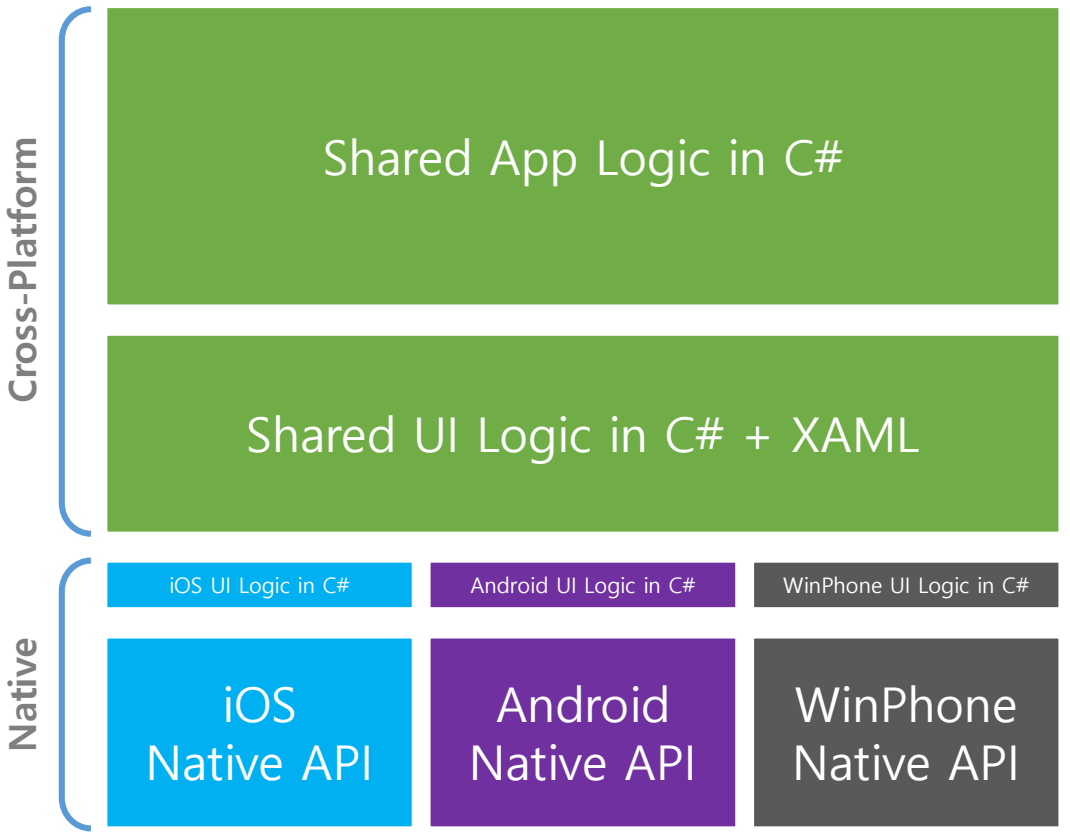
https://appsto.re/us/eJJ84.i

https://www.envicase.com/company/recruit



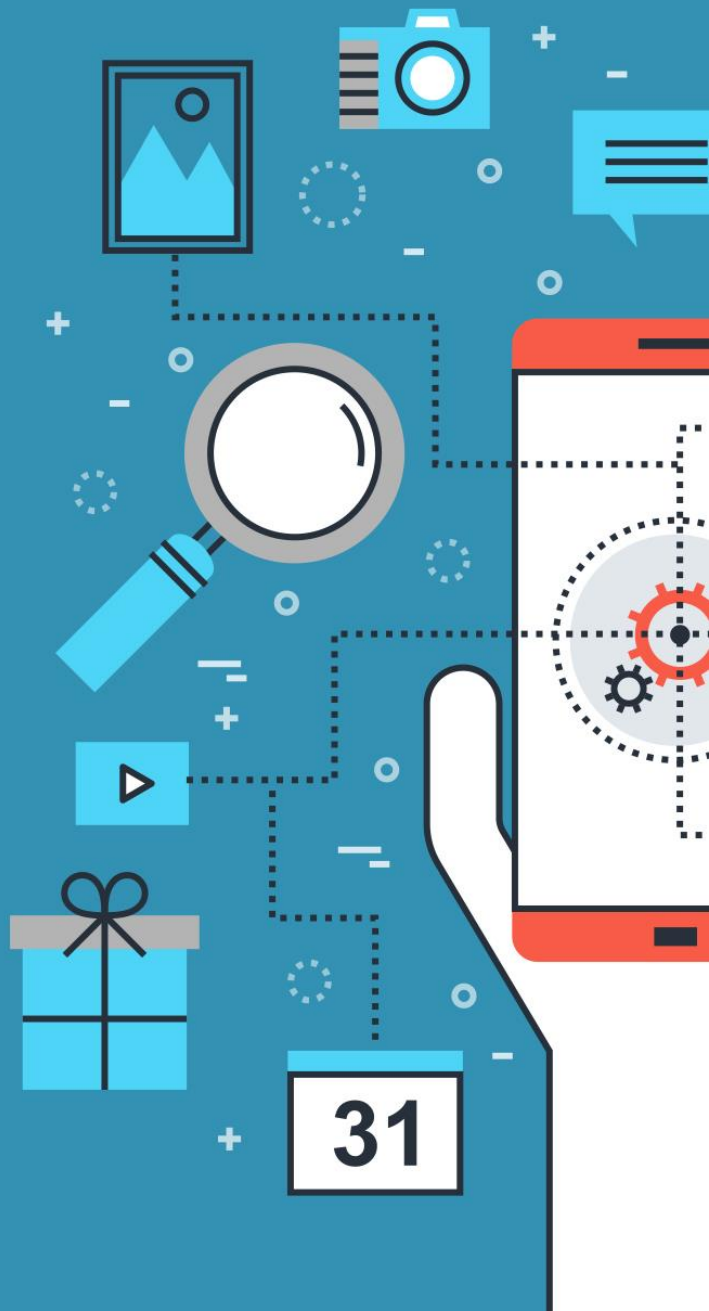
Xamarin Forms





내용

- 플랫폼 별 UI 작성
- 응용프로그램 모니터링
- 네이티브 서비스
- 신규 기능



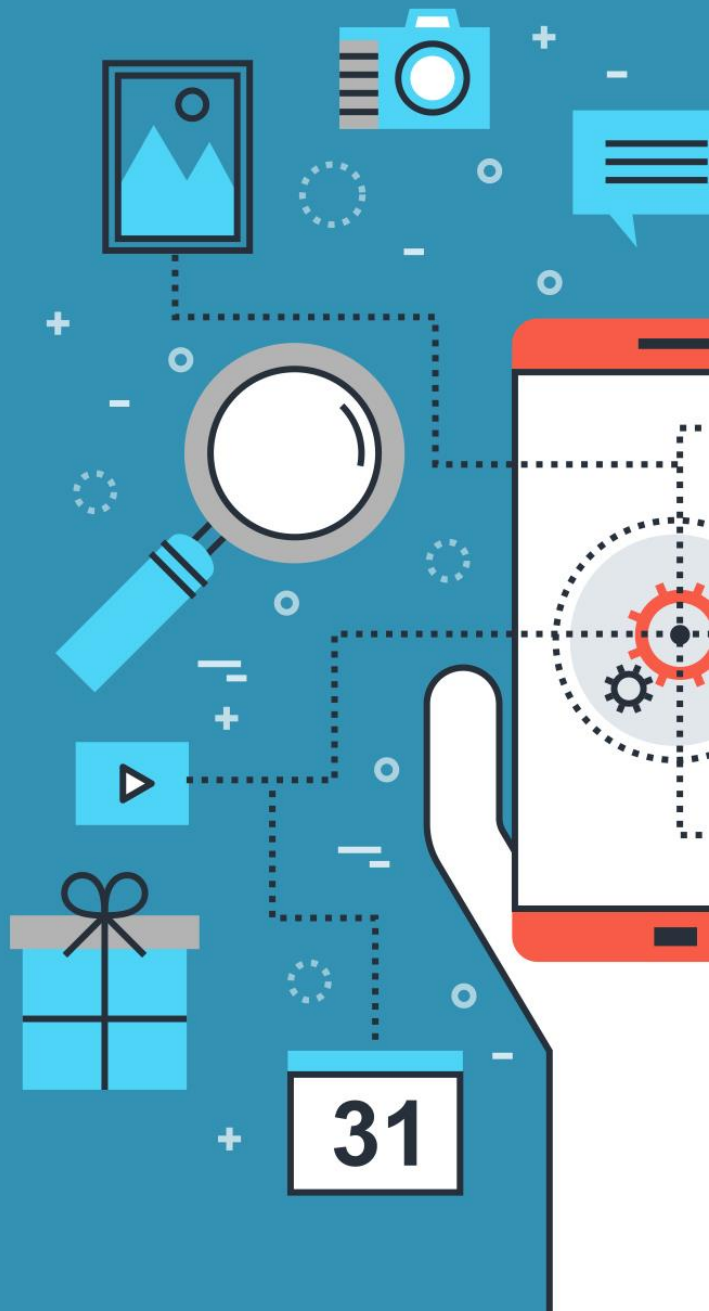
플랫폼 별 UI 작성

플랫폼별 UI 속성 설정

- 런타임에 플랫폼 별 값 제공

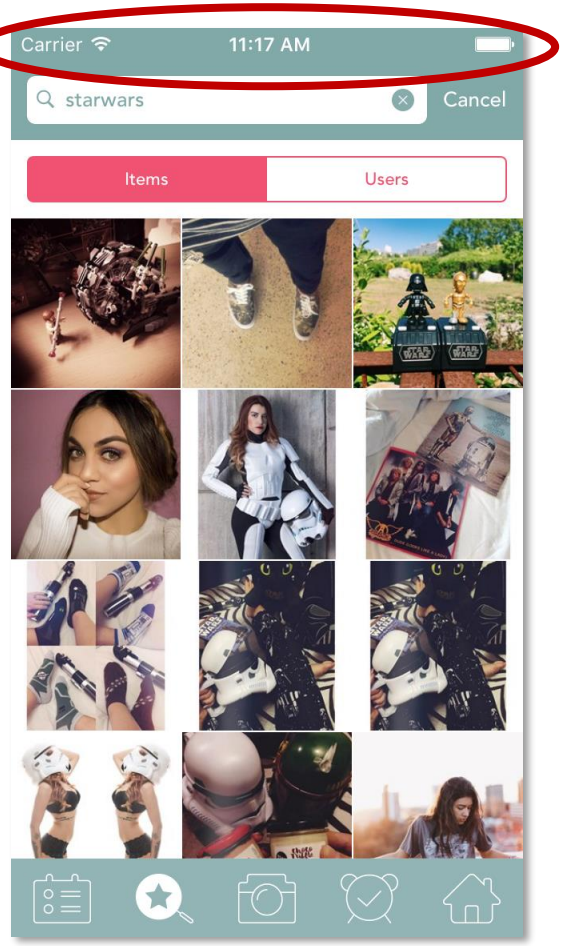
```
Device.OnPlatform<T>(
    T iOS, T Android, T WinPhone)
```

```
<OnPlatform x:TypeArguments="…"
            iOS="…"
            Android="…"
            WinPhone="…" />
```



<OnPlatform />

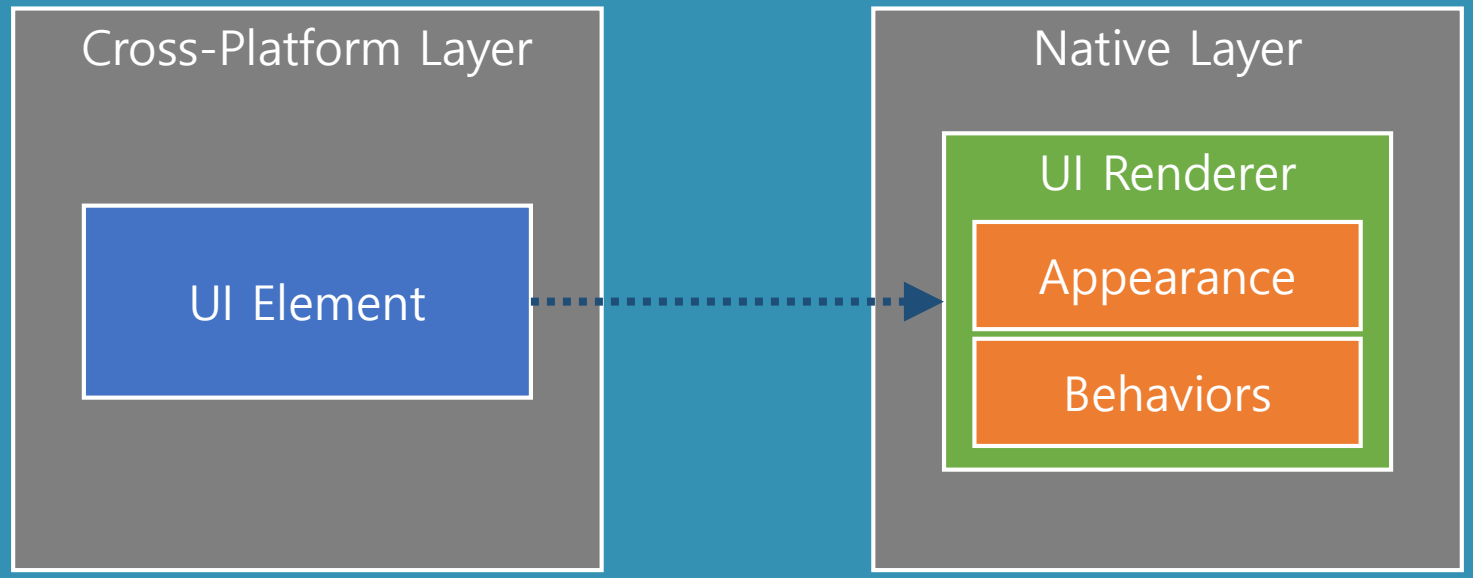

```
<ContentPage>
  <ContentPage.Padding>
    <OnPlatform x:TypeArguments="Thickness"
                iOS="0,20,0,0" />
  </ContentPage.Padding>
</ContentPage>
```



커스텀 렌더러

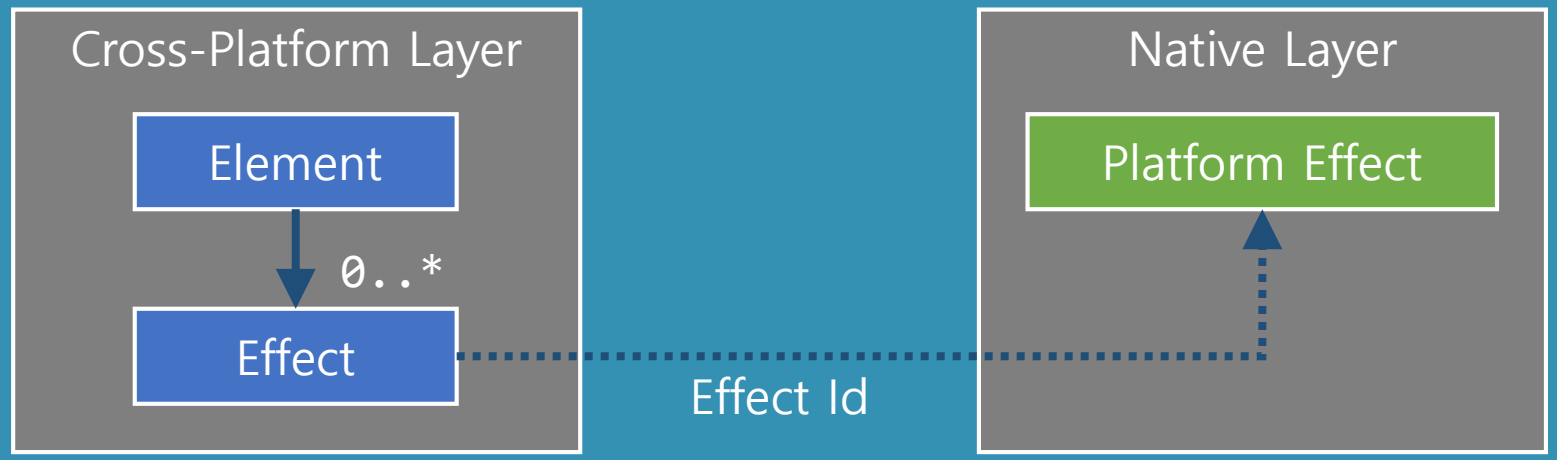
- 기존/ 신규 크로스 플랫폼 요소에 매핑
- Native API를 사용해 플랫폼 별 UI 요소의 외형과 동작 구현

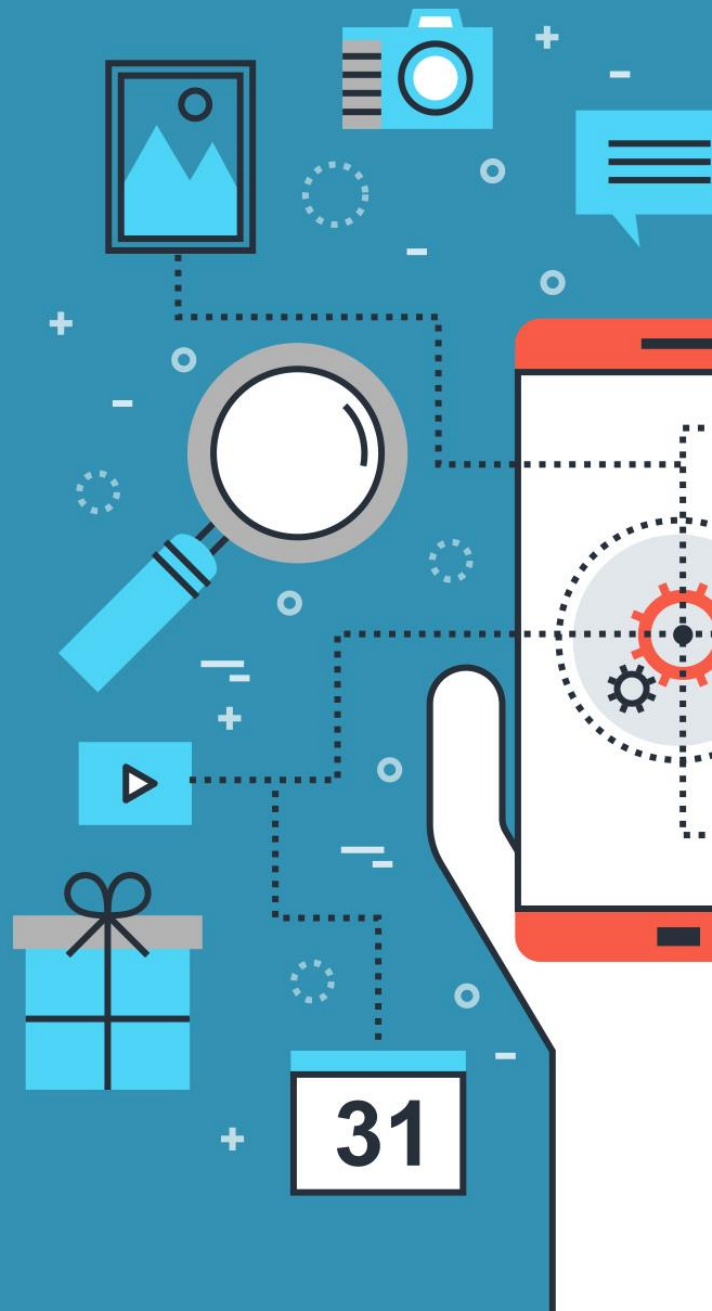
```
[assembly: ExportRenderer(typeof(UIElement), typeof(UIRenderer))]
```



효과(Effect)

- 네이티브 계층에서 효과 구현
- 크로스 플랫폼 계층에서 효과 적용(attach)/ 해제(detach)
- 간단한 UI 요소의 외형이나 동작을 변경
- Effect Id: *[ResolutionGroupName].[EffectName]*





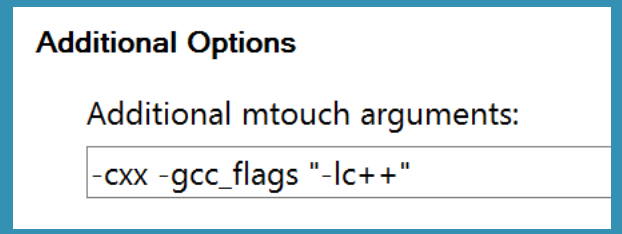
응용프로그램 모니터링

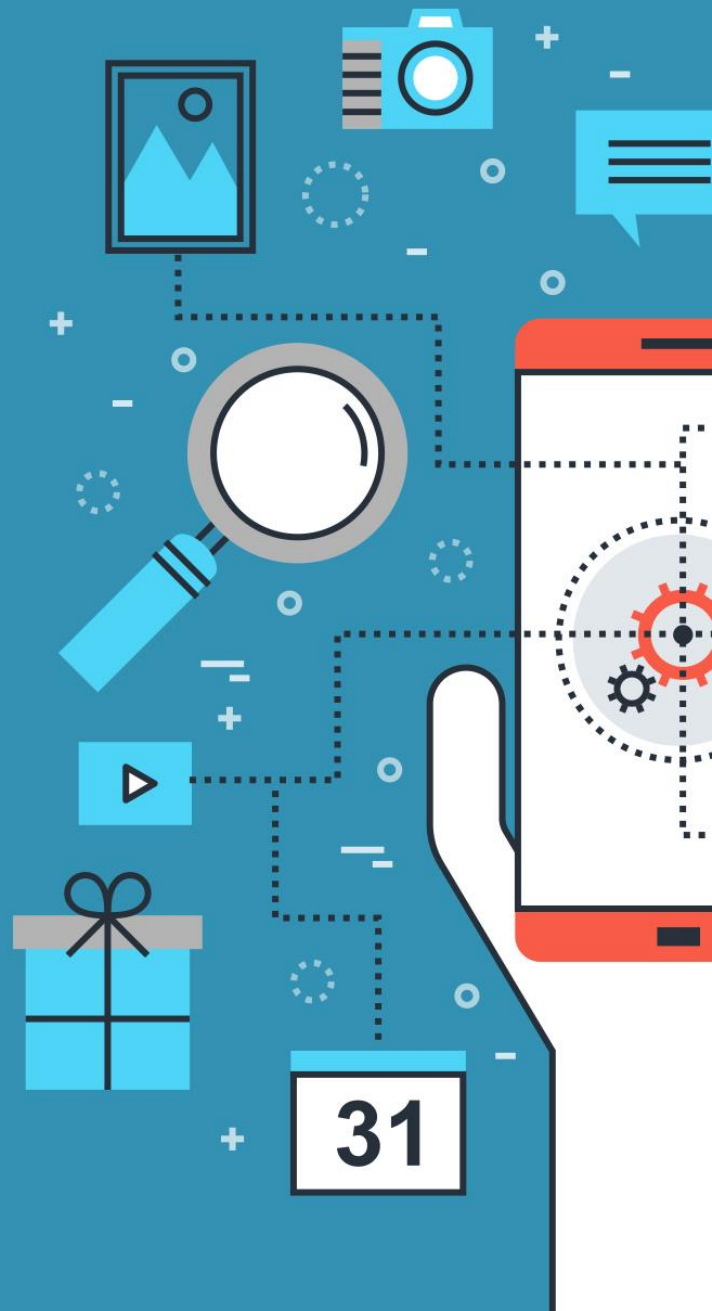
HockeyApp

- 모바일 응용프로그램 DevOps 플랫폼
- 베타 배포, 오류 수집, 사용자 데이터, 피드백, …
- 2014년 Microsoft가 인수
- Xamarin Insights가 HockeyApp으로 통합 진행 중
- https://www.hockeyapp.net/features/

HockeyApp for Xamarin

- Component 스토어
  https://components.xamarin.com/view/hockeyappios
  https://components.xamarin.com/view/hockeyappandroid
- iOS의 경우 빌드를 위해 mtouch 인수 추가 필요
  -cxx -gcc_flags "-lc++"



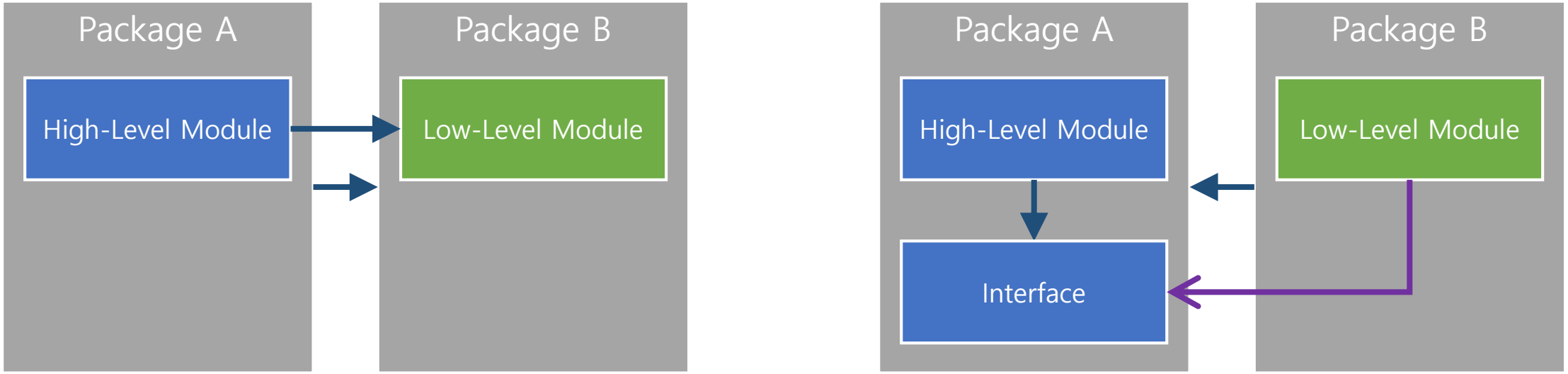


네이티브 서비스

의존성 역전 원리

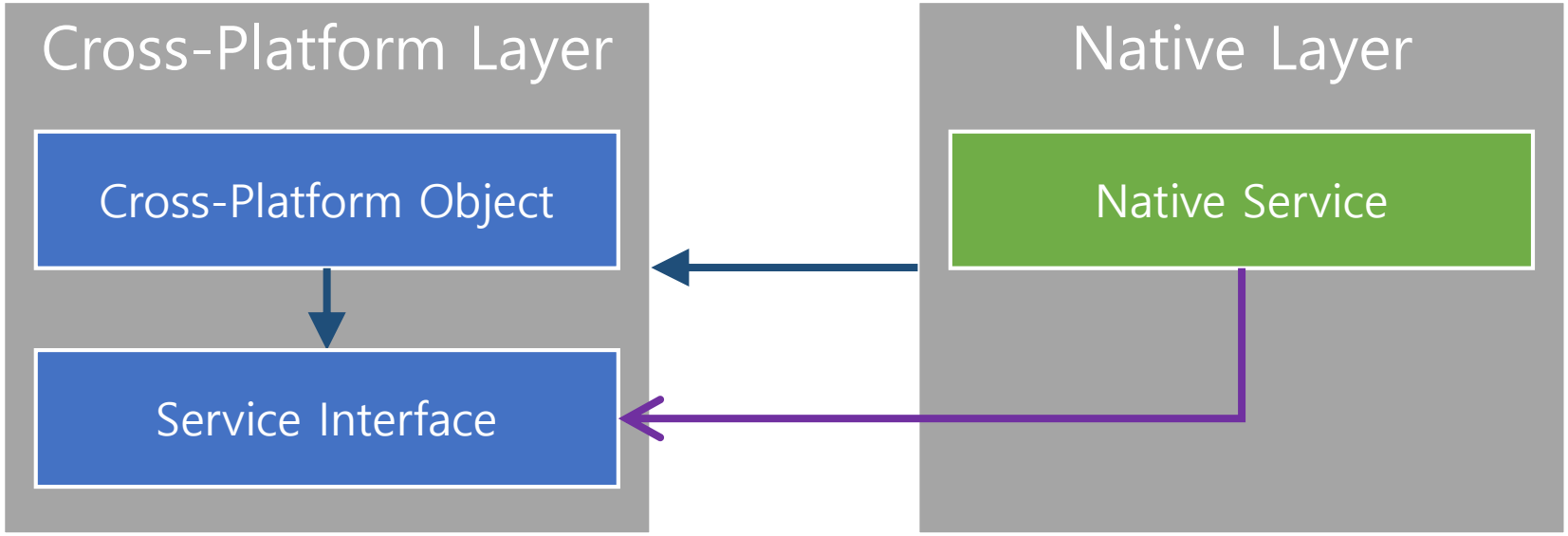
Dependency Inversion Principal

- *"높은 수준의 모듈은 낮은 수준의 모듈에 의존하지 않는다. 둘 모두는 추상화에 의존한다."*
- *"추상화는 세부사항에 의존하지 않는다. 세부사항이 추상화에 의존한다."*

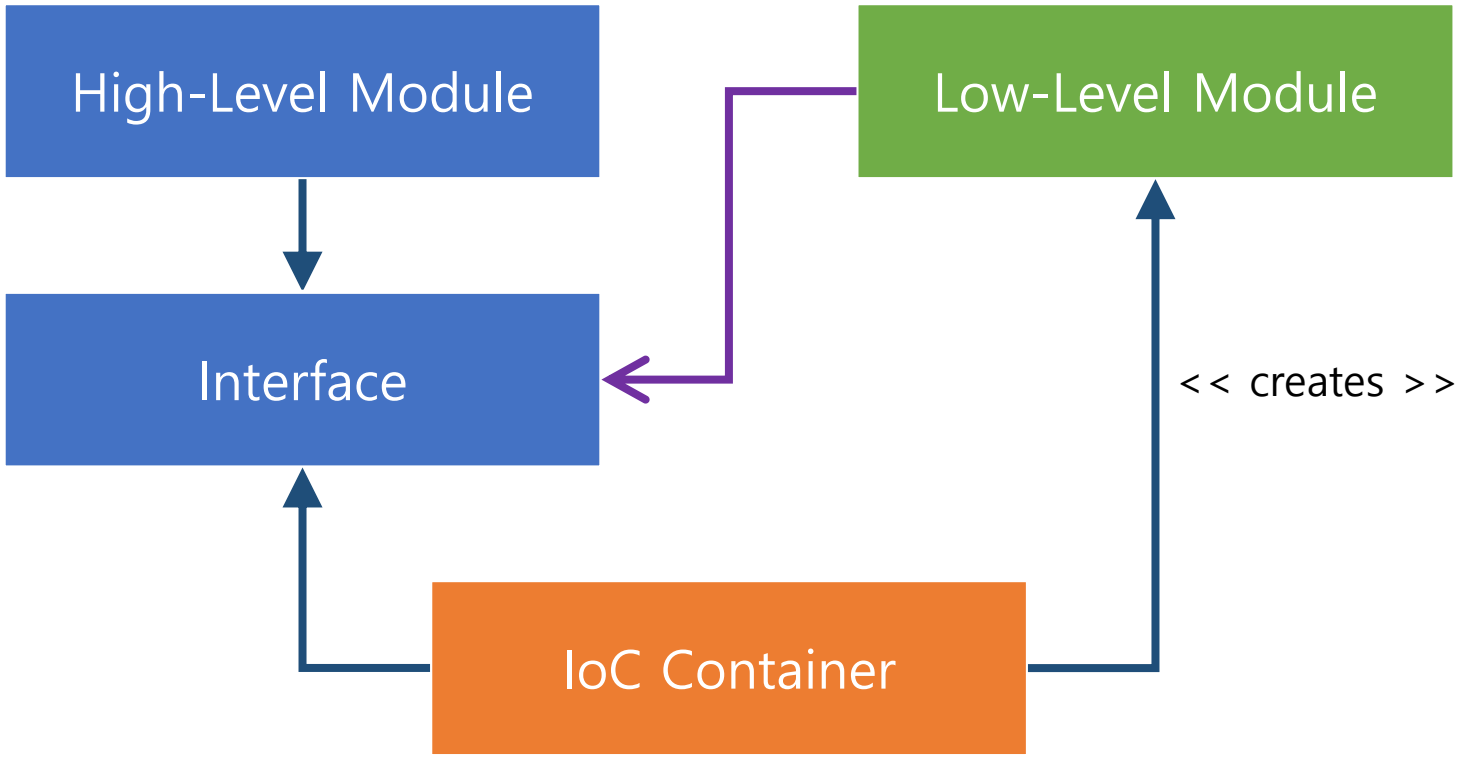


https://justhackem.wordpress.com/2016/05/13/dependency-inversion-terms/#dip

네이티브 서비스 사용


Cross-Platform Layer
Cross-Platform Object
Service Interface
Native Layer
Native Service

IoC 컨테이너

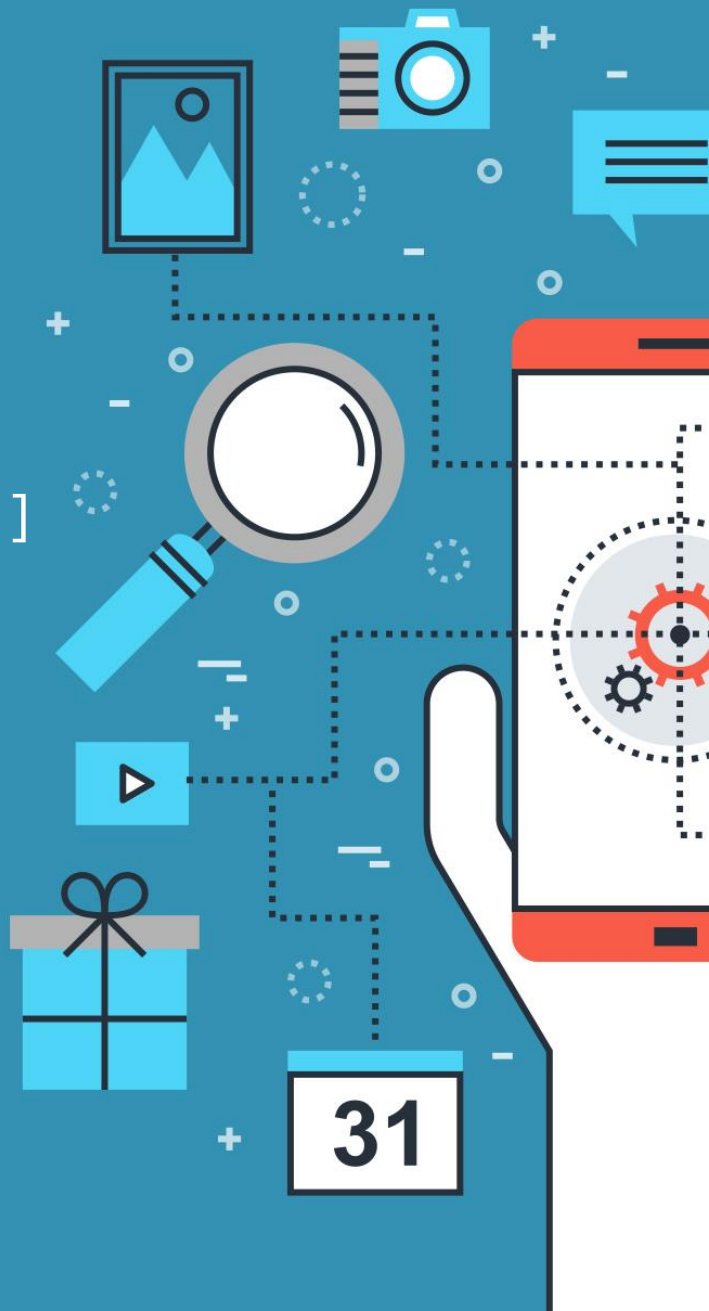


https://justhackem.wordpress.com/2016/05/13/dependency-inversion-terms/#ioc-container

DependencyService

- 단순한 Xamarin Forms 빌트 인 IoC 컨테이너
- 특성(attribute)를 사용한 서비스 구현체 등록
  `[assembly: Xamarin.Forms.Dependency(typeof(FooService))]`
- DependencyService 클래스를 통해 서비스 제공
  `DependencyService.Get<IFooService>()`

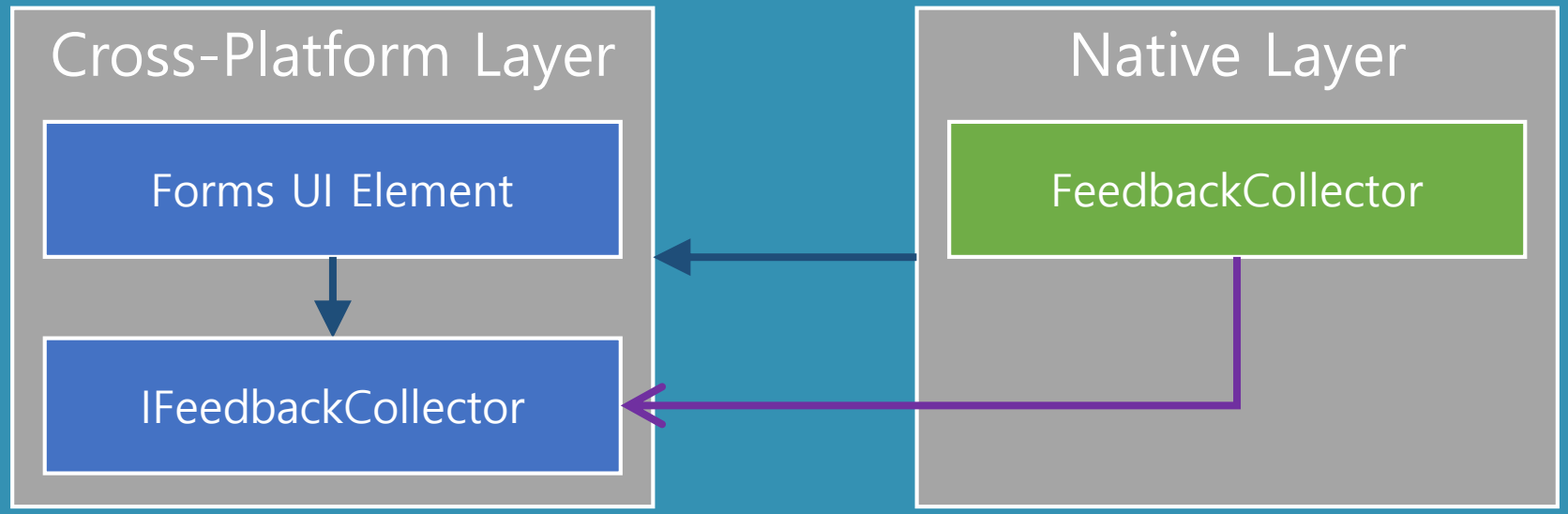
```
public class FooService : IFooService
{
}
```

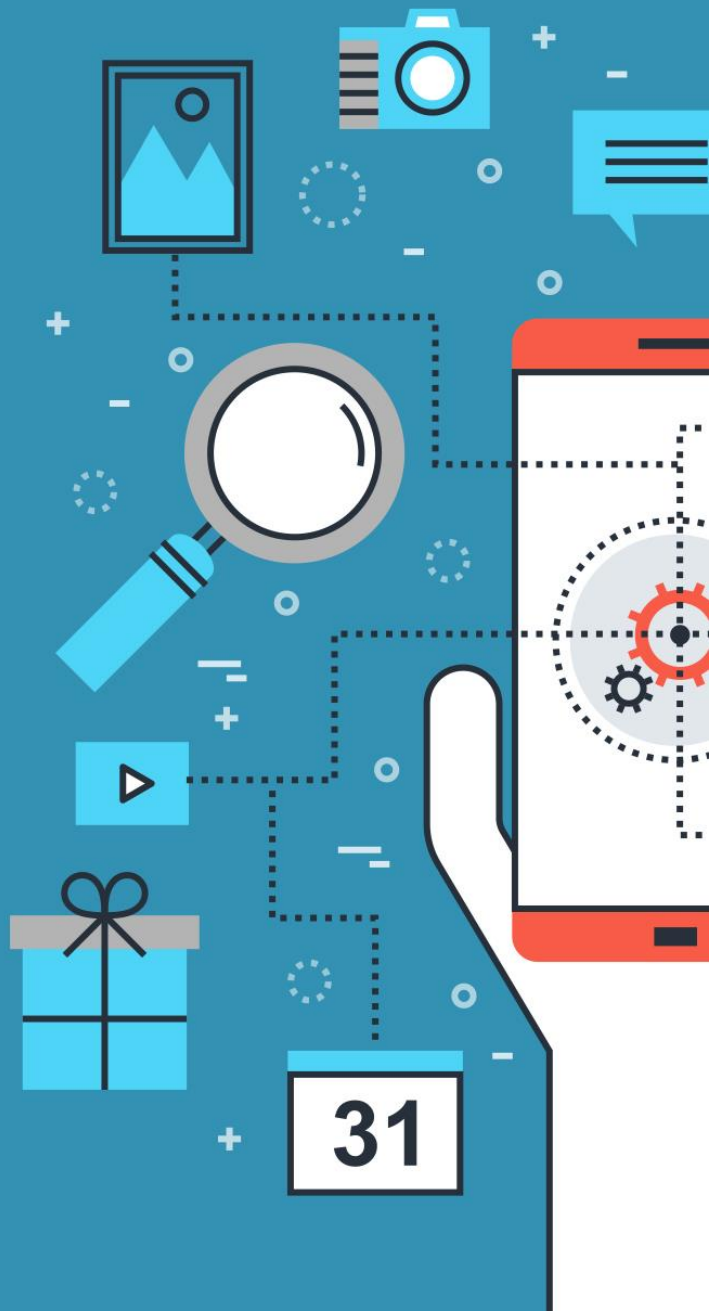


DependencyService 예제

HockeyApp 피드백 서비스

```
[assembly: Dependency(typeof(FeedbackCollector))]
DependencyService.Get<IFeedbackCollector>()
```





마무리

신규 기능

- Xamarin Inspector
  https://developer.xamarin.com/guides/cross-platform/inspector/
- Xamarin Workbooks
  https://developer.xamarin.com/guides/cross-platform/workbooks/
- SkiaSharp
  https://developer.xamarin.com/guides/cross-platform/drawing/
- iOS Simulator for Windows
  https://developer.xamarin.com/guides/cross-platform/windows/ios-simulator/