

 無料電子ブック

学習

xamarin

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

#xamarin

.....	1
<b>1: xamarin</b> .....	<b>2</b>
.....	2
Examples.....	2
OS XXamarin Studio.....	2
.....	4
.....	15
Xamarin StudioHello WorldXamarin.Forms.....	15
<b>2:</b> .....	<b>21</b>
.....	21
Examples.....	21
.....	21
<b>3:</b> .....	<b>23</b>
Examples.....	23
.....	23
.....	24
.....	<b>27</b>

---

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [xamarin](#)

It is an unofficial and free xamarin ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official xamarin.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)

---

# 1: xamarinをいめる

このセクションでは、xamarinのと、がxamarinをするについてをします。

それはまた、xamarinののきなし、トピックにリンクするがあります。 xamarinのドキュメントはしくなっているので、それらのトピックのバージョンをするがあります。

## Examples

### OS XでのXamarin Studioのインストール

OS XマシンでXamarinをするためののステップは、 [サイト](#)からXamarin Studio Communityをダウンロードしてインストールすることです。のにすように、インストーラをダウンロードするにはいくつかのフィールドをするがあります。



# Down

Nice! You are about to d

C# and sha

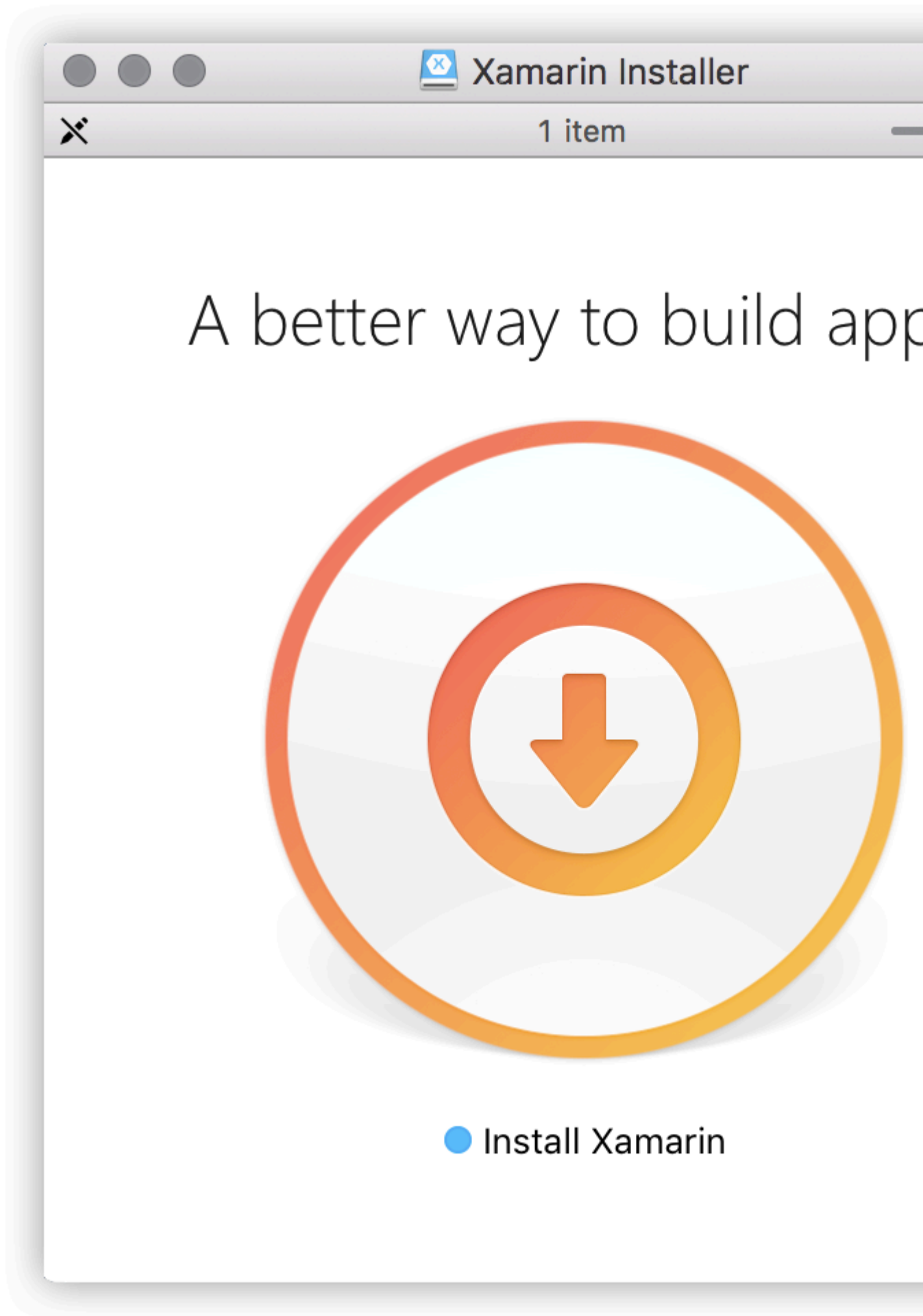
XamarinコンポーネントAndroid SDKなどをしてインストールします。

Xamarin.iOSアプリケーションをするには、のをたすがあります。

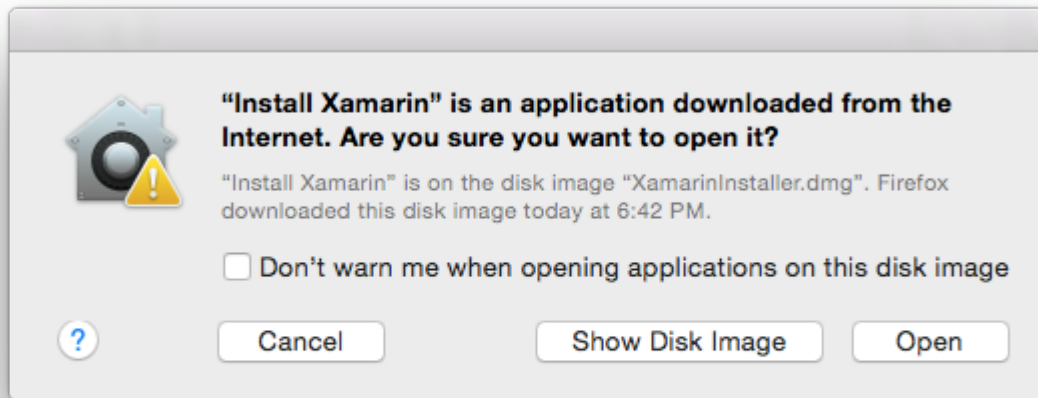
- [iOSデベロッパーセンター](#)ののiOS SDK。
- Mac App Storeまたは[Apple Developer Website](#)のXcodeのバージョン。
- Mac OS X Yosemite10.10

## インストールプロセス

がたされたら、XamarinロゴをダブルクリックしてXamarin Installerをします。



することがあります。するには「く」をクリックしてください。



のインストールプロセスをするには、Xamarinソフトウェアライセンスをみ、するがあります。 [にします]チェックボックスをオンにして、およびエラーをきめます。





- **Xamarin License**
- Prerequisites
- Installation
- Summary

## Read and accept

Xamarin Software License

Last updated: November

### 1. ACCEPTANCE

PLEASE READ THIS AGREEMENT FOR THE USE OF XAMARIN SOFTWARE (INCLUDING XAMARIN STUDIO) BEFORE YOU ACCEPT IT. IF YOU DO NOT AGREE

This Xamarin Software License is entered into by and between you (whether an individual or a person) and Xamarin Inc. ("Xamarin"), a company, an organization, or other entity. (i) You represent and warrant that you are an individual and you bind such entity to this Agreement on its behalf. For example, if you have executed a subscription agreement with Xamarin, you have agreed to the terms and conditions of this Agreement at <https://store.xamarin.com>. You have accepted this Agreement, electronically.

### 2. DEFINITIONS

I accept the terms of this Agreement

Automatically send me updates





インテル®HAXMはのにはみがないかもしれません。これは、インテル®ハードウェアアクセラレーションマネージャのであり、Androidエミュレーションをします。

すでにシステムにインストールされているはされていますが、グレーされています。



- Xamarin License
- **Product Selection**
- Configuration
- Prerequisites
- Installation
- Summary

## Please select product

-  Xamarin
-  Xamarin
-  Xamarin
-  Intel®

ステップでXamarin.Androidがされているは、Android SDKのインストールをするようめられます。ほとんどの、デフォルトのはなですので、「ける」をしてしてください。



- Xamarin License
- Product Selection
- **Configuration**
- Prerequisites
- Installation
- Summary

## Please configure



/Users/hankidesign/Li

Private copy of Android

Size on disk: **338.7**

Download size: **948.3**

Xamarin.Androidはまだインストールされていないため、のとともにリストにされます。

☐をクリックすると、のダウンロードとインストールのプロセスがされます。インストーラは、のシステムユーザのユーザとパスワードをすするダイアログをすることによって、システムをすするをすることがあります。をし、「OK」をクリックしてインストールをめてください。



- Xamarin License
- Product Selection
- Configuration
- Prerequisites
- **Installation**
- Summary

## Installation in progress

Sit back and relax - we'll take care of the rest.



**Mono Framework**

4.4.1

Downloading Mono Framework  
Downloaded 170.77MB (current)

はありませんが、エンタープライズをするにはアカウントをし、をにするがあります。





- Xamarin License
- Product Selection
- Configuration
- Prerequisites
- Installation
- **Summary**

## Installation Finished

### Installation

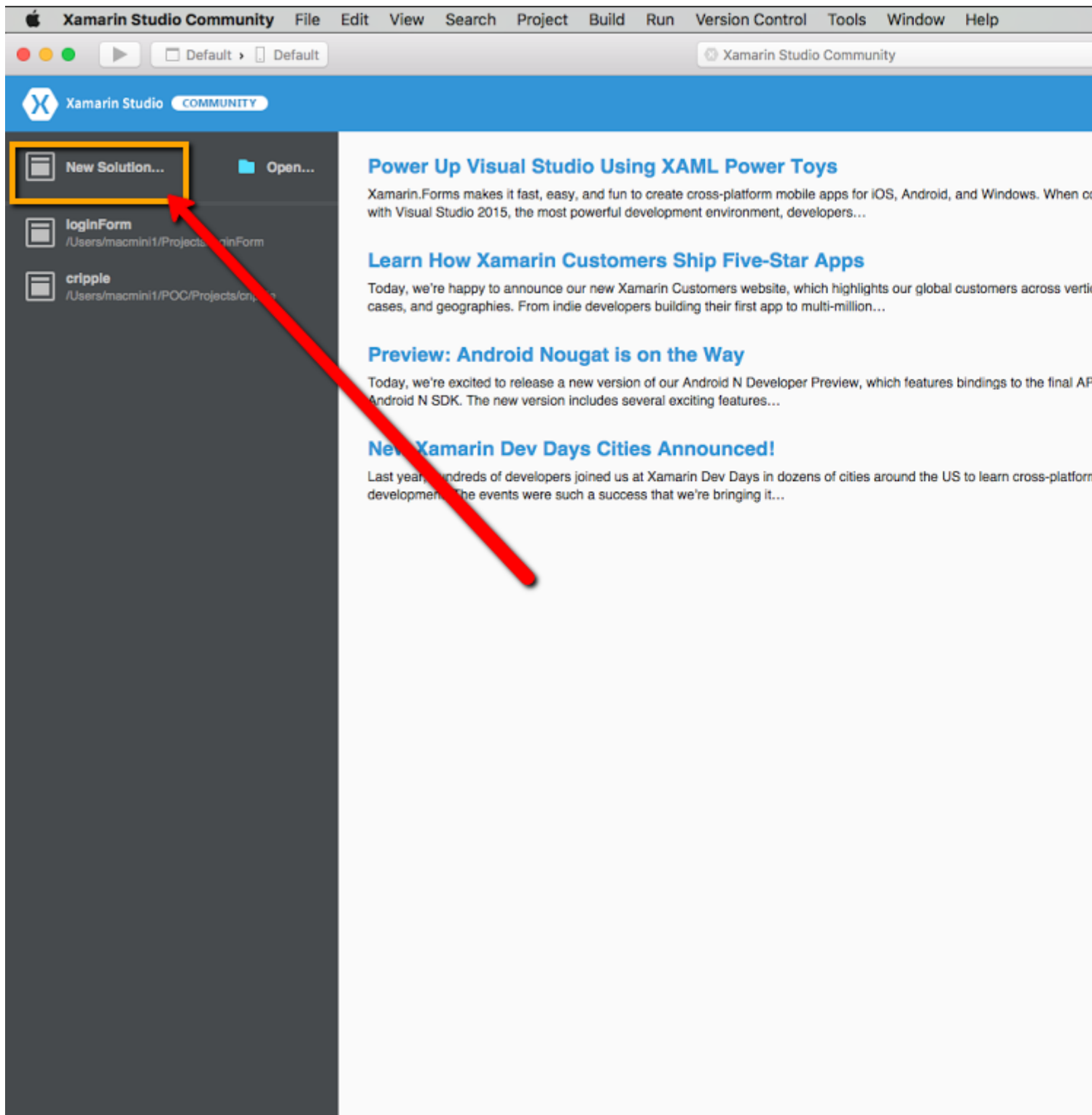
Now you are ready to use  
Visual Studio

するための新しいライブラリです。ランタイムにネイティブコントロールにマッピングされる40のクロスプラットフォームのコントロールとレイアウトをします。つまり、ユーザーインターフェイスはネイティブです

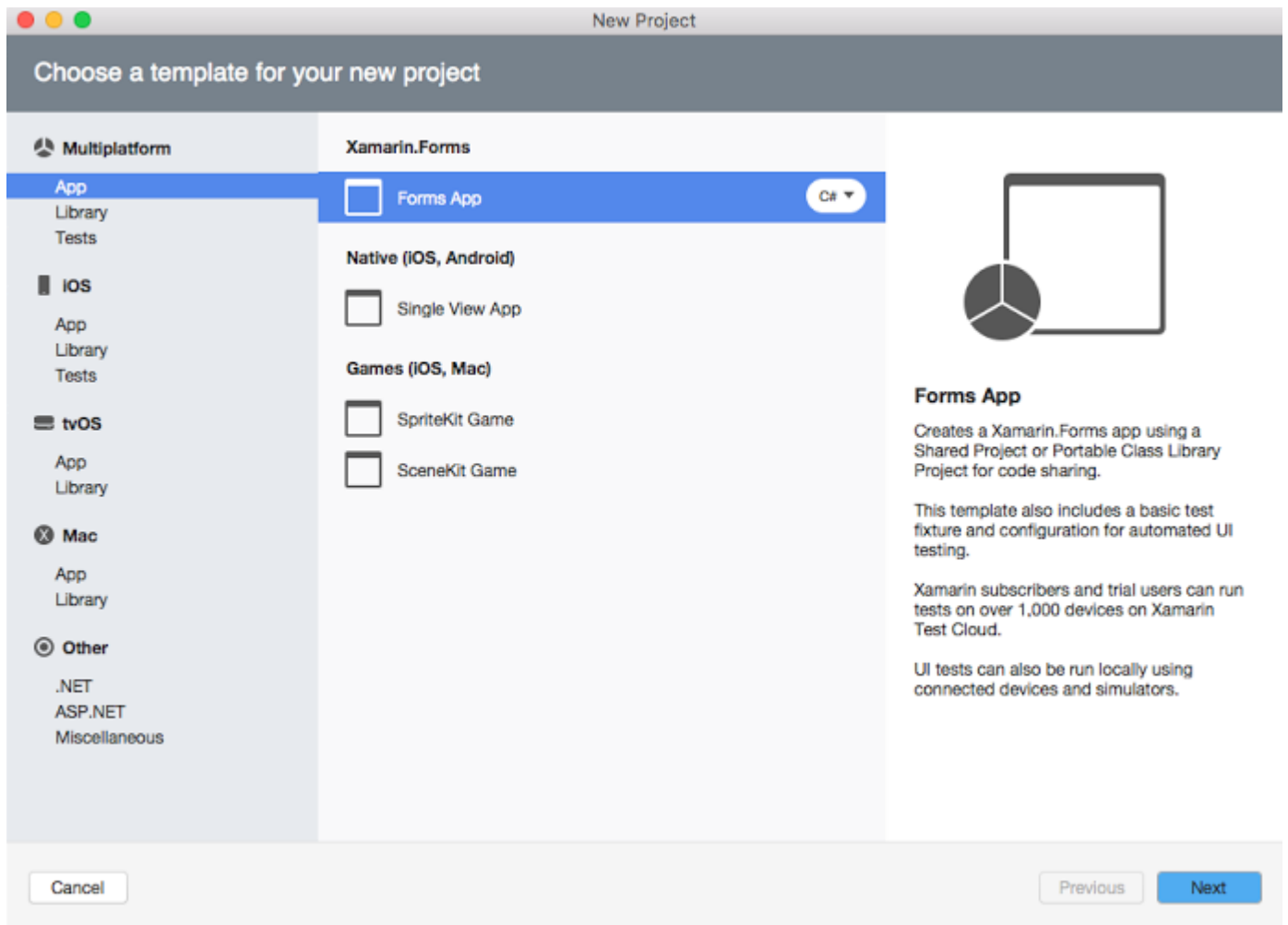
## ステップ1

新しいソリューションをします。

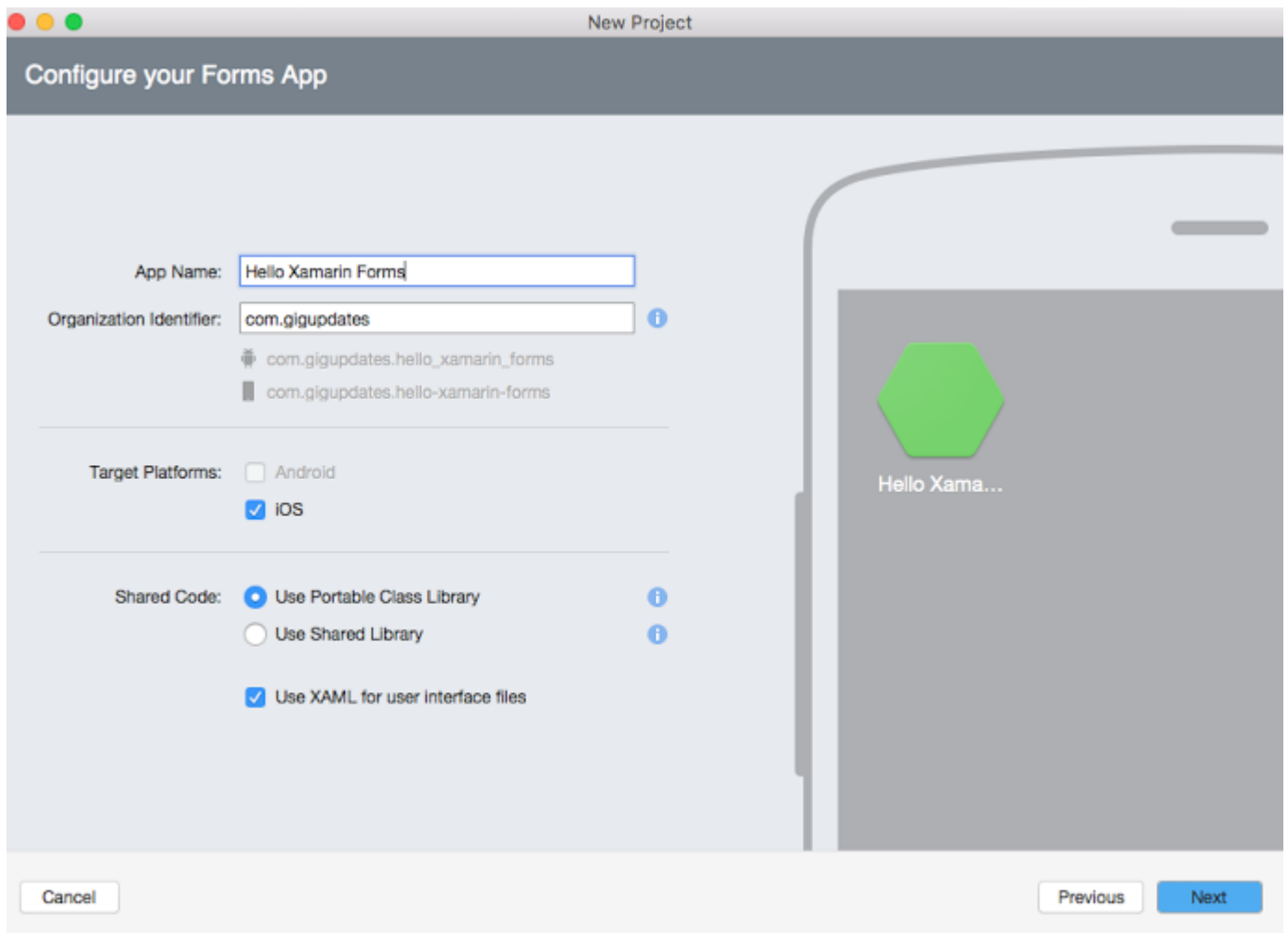
「新しいソリューション」をクリックします。



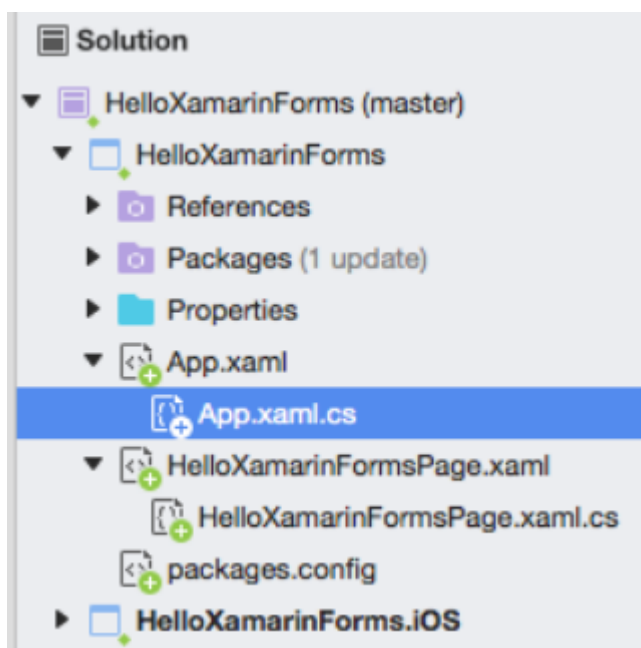
ステップ2フォームアプリケーションをし、へをクリックします。



ステップ3アプリケーションをし、へをクリックします。



これは、ソリューションがされたときのようにプロジェクトのがどのようにえるかをしています。



## App.xaml

```
<?xml version="1.0" encoding="utf-8"?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
```

```

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="HelloXamarinForms.App">
<Application.Resources>
  <!-- Application resource dictionary -->
</Application.Resources>
</Application>

```

## App.xaml.cs

```

using Xamarin.Forms;

namespace HelloXamarinForms
{
    public partial class App : Application
    {
        public App()
        {
            InitializeComponent();

            MainPage = new HelloXamarinFormsPage();
        }

        protected override void OnStart()
        {
            // Handle when your app starts
        }

        protected override void OnSleep()
        {
            // Handle when your app sleeps
        }

        protected override void OnResume()
        {
            // Handle when your app resumes
        }
    }
}

```

## HelloXamarinFormsPage.xaml

```

<?xml version="1.0" encoding="utf-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  xmlns:local="clr-namespace:HelloXamarinForms"
  x:Class="HelloXamarinForms.HelloXamarinFormsPage">

  <Label Text="Welcome to Xamarin Forms!" VerticalOptions="Center"
    HorizontalOptions="Center" />
</ContentPage>

```

## HelloXamarinFormsPage.xaml.cs

```

using Xamarin.Forms;

namespace HelloXamarinForms
{

```

```
public partial class HelloXamarinFormsPage : ContentPage
{
    public HelloXamarinFormsPage()
    {
        InitializeComponent();
    }
}
```

オンラインでxamarinをいめるをむ <https://riptutorial.com/ja/xamarin/topic/899/xamarinをいめる>

## 2: アノテーションによるオブジェクトの

き

mvc.netは、モデルのためのデータアノテーションをしています。これはXamarinでもうことができます

### Examples

な

ナゲットパッケージ `System.ComponentModel.Annotations` する

クラスをする

```
public class BankAccount
{
    public enum AccountType
    {
        Saving,
        Current
    }

    [Required(ErrorMessage="First Name Required")]
    [MaxLength(15,ErrorMessage="First Name should not more than 1`5 character")]
    [MinLength(3,ErrorMessage="First Name should be more than 3 character")]
    public string AccountHolderFirstName { get; set; }

    [Required(ErrorMessage="Last Name Required")]
    [MaxLength(15,ErrorMessage="Last Name should not more than 1`5 character")]
    [MinLength(3,ErrorMessage="Last Name should be more than 3 character")]
    public string AccountHolderLastName { get; set; }

    [Required]
    [RegularExpression("[0-9]+$", ErrorMessage = "Only Number allowed in AccountNumber")]
    public string AccountNumber { get; set; }

    public AccountType AcType { get; set; }
}
```

バリデータをする

```
public class GenericValidator
{
    public static bool TryValidate(object obj, out ICollection<ValidationResult> results)
    {
        var context = new ValidationContext(obj, serviceProvider: null, items: null);
        results = new List<ValidationResult>();
        return Validator.TryValidateObject(
            obj, context, results,
```

```
        validateAllProperties: true
    );
}
}
```

バリデーターをしてください

```
var bankAccount = new BankAccount();
ICollection<ValidationResult> lstvalidationResult;

bool valid = GenericValidator.TryValidate(bankAccount, out lstvalidationResult);
if (!valid)
{
    foreach (ValidationResult res in lstvalidationResult)
    {
        Console.WriteLine(res.MemberNames + ":" + res.ErrorMessage);
    }
}

Console.ReadLine();
```

された

```
First Name Required
Last Name Required
The AccountNumber field is required.
```

オンラインでアノテーションによるオブジェクトのをむ

<https://riptutorial.com/ja/xamarin/topic/9720/アノテーションによるオブジェクトの>



## 3: プロジェクトのコード

### Examples

#### ブリッジパターン

ブリッジパターンは、もなInversion of Controlデザインパターンの1つです。 Xamarinの、このパターンはプラットフォームにしないコンテキストからプラットフォームコードをするためにされます。たとえば、ポータブルクラスライブラリやXamarinフォームのAndroidのAlertDialogをします。これらのコンテキストのどちらも、AlertDialogオブジェクトがであるかはかっていませんので、するためにはボックスにラップする必要があります。

```
// Define a common interface for the behavior you want in your common project (Forms/Other PCL)
public interface IPlatformReporter
{
    string GetPlatform();
}

// In Android/iOS/Win implement the interface on a class
public class DroidReporter : IPlatformReporter
{
    public string GetPlatform()
    {
        return "Android";
    }
}

public class IosReporter : IPlatformReporter
{
    public string GetPlatform()
    {
        return "iOS";
    }
}

// In your common project (Forms/Other PCL), create a common class to wrap the native implementations
public class PlatformReporter : IPlatformReporter
{
    // A function to get your native implementation
    public static func<IPlatformReporter> GetReporter;

    // Your native implementation
    private IPlatformReporter _reporter;

    // Constructor accepts native class and stores it
    public PlatformReporter(IPlatformReporter reporter)
    {
        _reporter = GetReporter();
    }
}
```

```

// Implement interface behavior by deferring to native class
public string GetPlatform()
{
    return _reporter.GetPlatform();
}
}

// In your native code (probably MainActivity/AppDelegate), you just supply a function that
returns your native implementation
public class MainActivity : Activity
{
    protected override void OnCreate(Bundle bundle)
    {
        base.OnCreate(bundle);
        SetContentView(Resource.Layout.activity_main);

        PlatformReporter.GetReporter = () => { return new DroidReporter(); };
    }
}

public partial class AppDelegate : UIApplicationDelegate
{
    UIWindow window;

    public override bool FinishedLaunching(UIApplication app, NSDictionary options)
    {
        window = new UIWindow(UIScreen.MainScreen.Bounds);
        window.RootViewController = new UIViewController();
        window.MakeKeyAndVisible();

        PlatformReporter.GetReporter = () => { return new IosReporter(); };

        return true;
    }
}

// When you want to use your native implementation in your common code, just do as follows:
public void SomeFuncWhoCares()
{
    // Some code here...

    var reporter = new PlatformReporter();
    string platform = reporter.GetPlatform();

    // Some more code here...
}

```

## サービスロケータパターン

サービスロケータのパターンは、[この](#)にのいです。ブリッジパターンとに、このパターンはプラットフォームにしないコンテキストからプラットフォームコードをするためにできます。もいことに、このパターンはシングルトンパターンにしています。サービスロケータにれたものはすべてデファクトシングルトンになります。

```

// Define a service locator class in your common project
public class ServiceLocator {
    // A dictionary to map common interfaces to native implementations
    private Dictionary<object, object> _services;

    // A static instance of our locator (this guy is a singleton)
    private static ServiceLocator _instance;

    // A private constructor to enforce the singleton
    private ServiceLocator() {
        _services = new Dictionary<object, object>();
    }

    // A Singleton access method
    public static ServiceLocator GetInstance() {
        if(_instance == null) {
            _instance = new ServiceLocator();
        }

        return _instance;
    }

    // A method for native projects to register their native implementations against the
    common interfaces
    public static void Register(object type, object implementation) {
        _services?.Add(type, implementation);
    }

    // A method to get the implementation for a given interface
    public static T Resolve<T>() {
        try {
            return (T) _services[typeof(T)];
        } catch {
            throw new ApplicationException($"Failed to resolve type: {typeof(T).FullName}");
        }
    }

    //For each native implementation, you must create an interface, and the native classes
    implementing that interface
    public interface IA {
        int DoAThing();
    }

    public interface IB {
        bool IsMagnificent();
    }

    public class IosA : IA {
        public int DoAThing() {
            return 5;
        }
    }

    public class DroidA : IA {
        public int DoAThing() {
            return 42;
        }
    }

```

```

}

// You get the idea...

// Then in your native initialization, you have to register your classes to their interfaces
like so:
public class MainActivity : Activity
{
    protected override void OnCreate(Bundle bundle)
    {
        base.OnCreate(bundle);
        SetContentView(Resource.Layout.activity_main);

        var locator = ServiceLocator.GetInstance();
        locator.Register(typeof(IA), new DroidA());
        locator.Register(typeof(IB), new DroidB());
    }
}

public partial class AppDelegate : UIApplicationDelegate
{
    UIWindow window;

    public override bool FinishedLaunching(UIApplication app, NSDictionary options)
    {
        window = new UIWindow(UIScreen.MainScreen.Bounds);
        window.RootViewController = new UIViewController();
        window.MakeKeyAndVisible();

        var locator = ServiceLocator.GetInstance();
        locator.Register(typeof(IA), new IosA());
        locator.Register(typeof(IB), new IosB());

        return true;
    }
}

// Finally, to use your native implementations from non-native code, do as follows:
public void SomeMethodUsingNativeCodeFromNonNativeContext() {
    // Some boring code here

    // Grabbing our native implementations for the current platform
    var locator = ServiceLocator.GetInstance();
    IA myIA = locator.Resolve<IA>();
    IB myIB = locator.Resolve<IB>();

    // Method goes on to use our fancy native classes
}

```

オンラインでプロジェクトのコードをむ <https://riptutorial.com/ja/xamarin/topic/6183/プロジェクトのコード>

## クレジット

S. No		Contributors
1	xamarinをいめる	<a href="#">Akshay Kulkarni</a> , <a href="#">Community</a> , <a href="#">Gil Sand</a> , <a href="#">hankide</a> , <a href="#">Joel Martinez</a> , <a href="#">Marius Ungureanu</a> , <a href="#">Sven-Michael Stübe</a> , <a href="#">thomasvdb</a>
2	アノテーションによるオブジェクトの	<a href="#">Niek de Gooijer</a>
3	プロジェクトのコード	<a href="#">kellen lask</a> , <a href="#">valdetero</a>