Бесплатная электронная книга

учусь

xamarin

Free unaffiliated eBook created from
Stack Overflow contributors.

#xamarin

Около

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

глава 1: Начало работы с xamarin

замечания

В этом разделе представлен обзор того, что такое xamarin, и почему разработчик может захотеть его использовать.

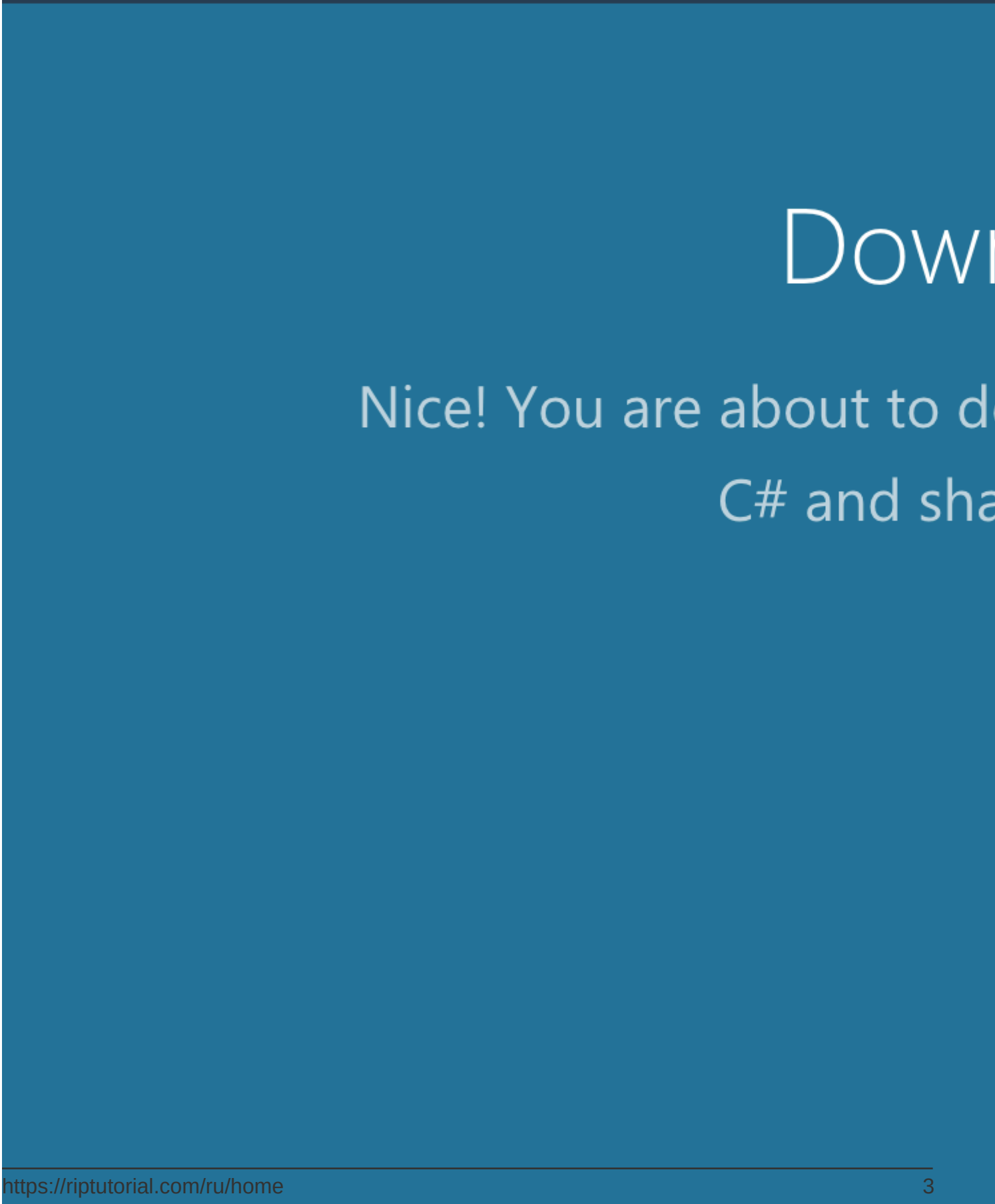
Следует также упомянуть о любых крупных предметах в рамках xamarin и ссылки на связанные темы. Поскольку документация для xamarin является новой, вам может потребоваться создать начальные версии этих связанных тем.

Examples

Установка Xamarin Studio на OS X

Первым шагом для начала разработки Xamarin на машине OS X является загрузка и установка версии Xamarin Studio Community с официального сайта . Для загрузки установщика необходимо заполнить несколько полей, как показано на рисунке ниже.


Xamarin
Products
Custom

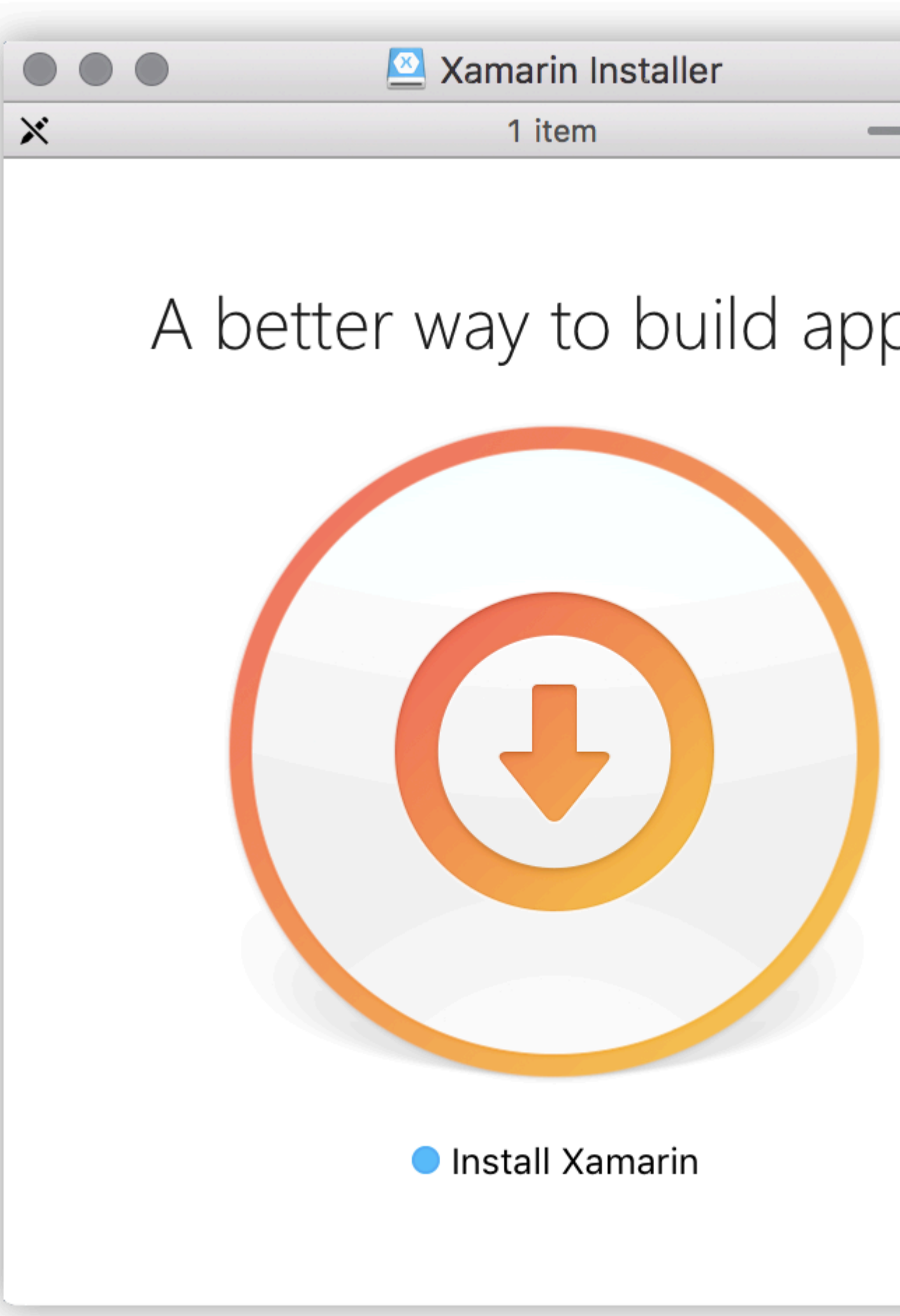

Down
Nice! You are about to d
C# and sha

заботится об идентификации и установке всех необходимых компонентов, отличных от Xamarin (например, Android SDK) поверх Xamarin.Android, Xamarin.iOS и Xamarin Studio.

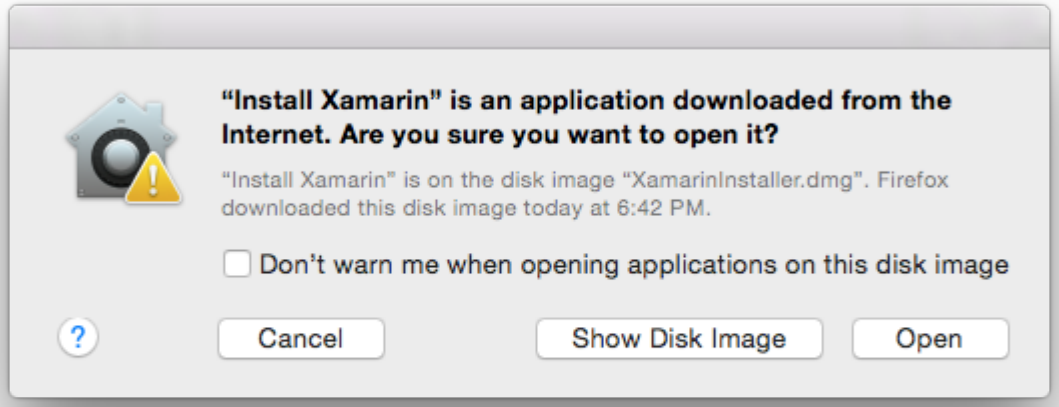
> Для разработки приложений Xamarin.iOS необходимо выполнить следующие предпосылки:
>
> - Последний iOS SDK от разработчика iOS .
> - Последняя версия Xcode из Mac App Store или веб-сайта разработчика Apple .
> - Mac OS X Yosemite (10.10) и выше

Процесс установки

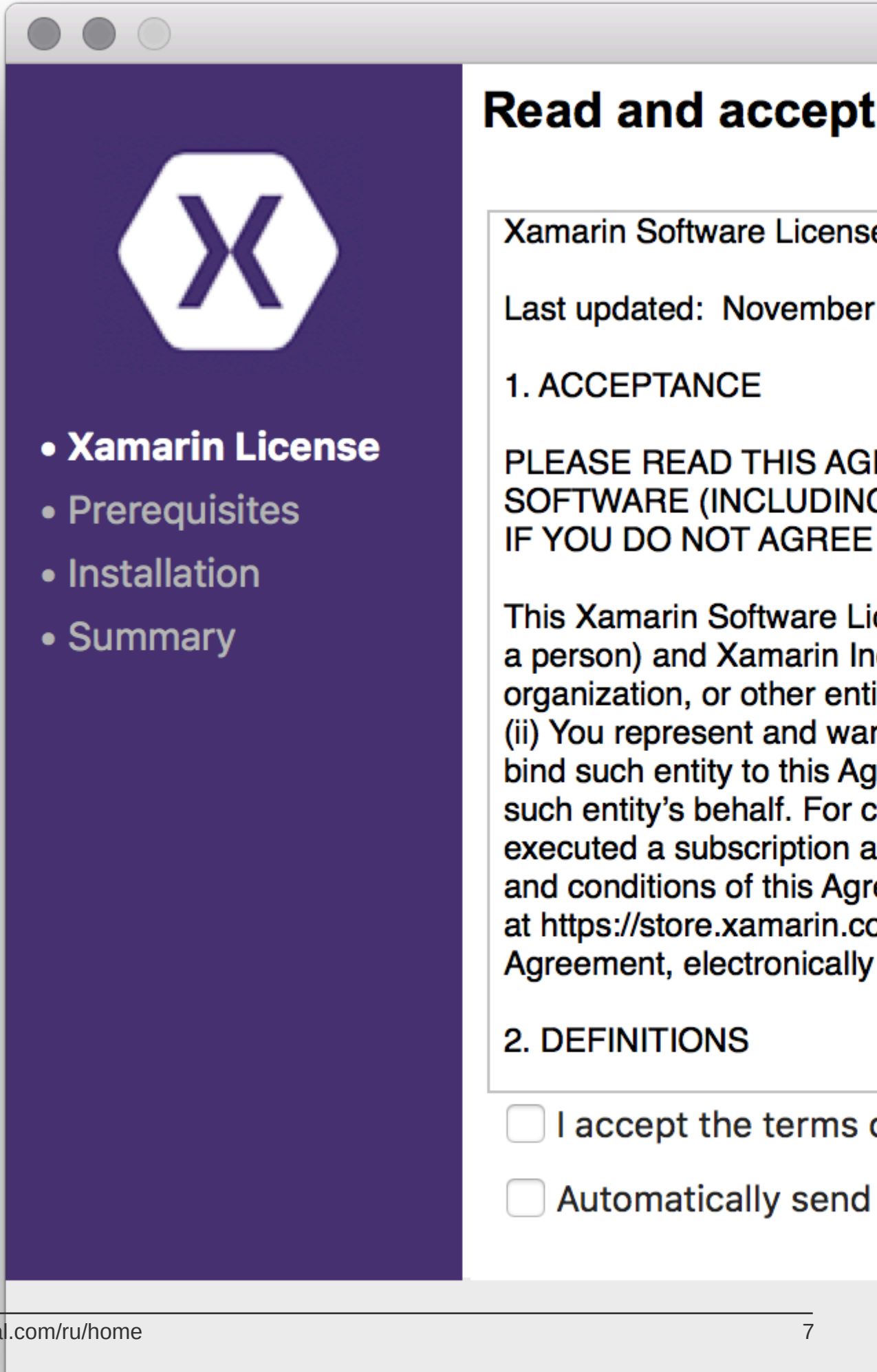
После того, как предварительные условия были выполнены, запустите установщик Xamarin, дважды щелкнув логотип Xamarin.


Xamarin Installer
1 item
A better way to build app
Install Xamarin

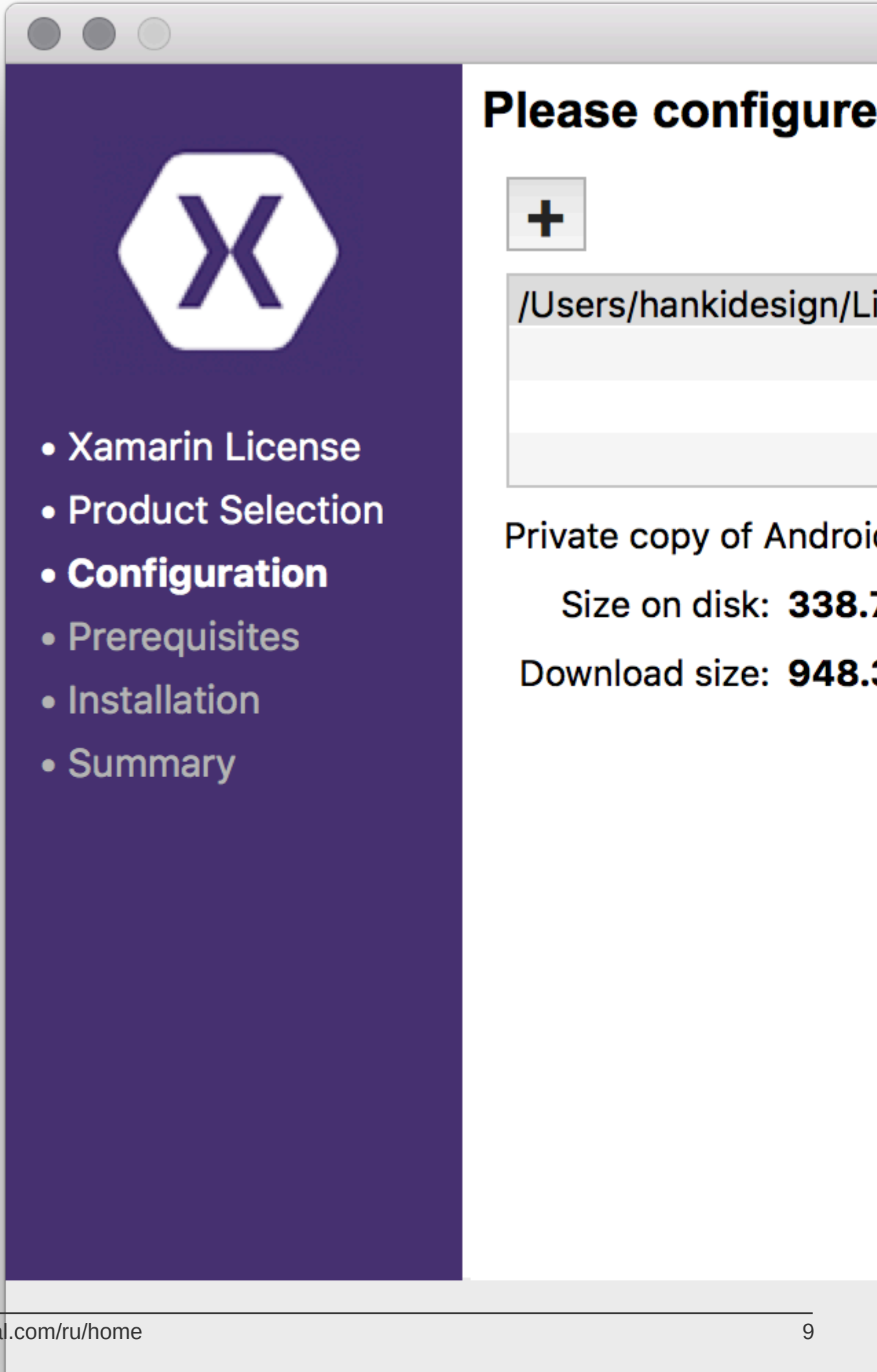
может отображать диалоговое окно с запросом подтверждения открытия загруженного приложения. Нажмите «открыть», чтобы продолжить.



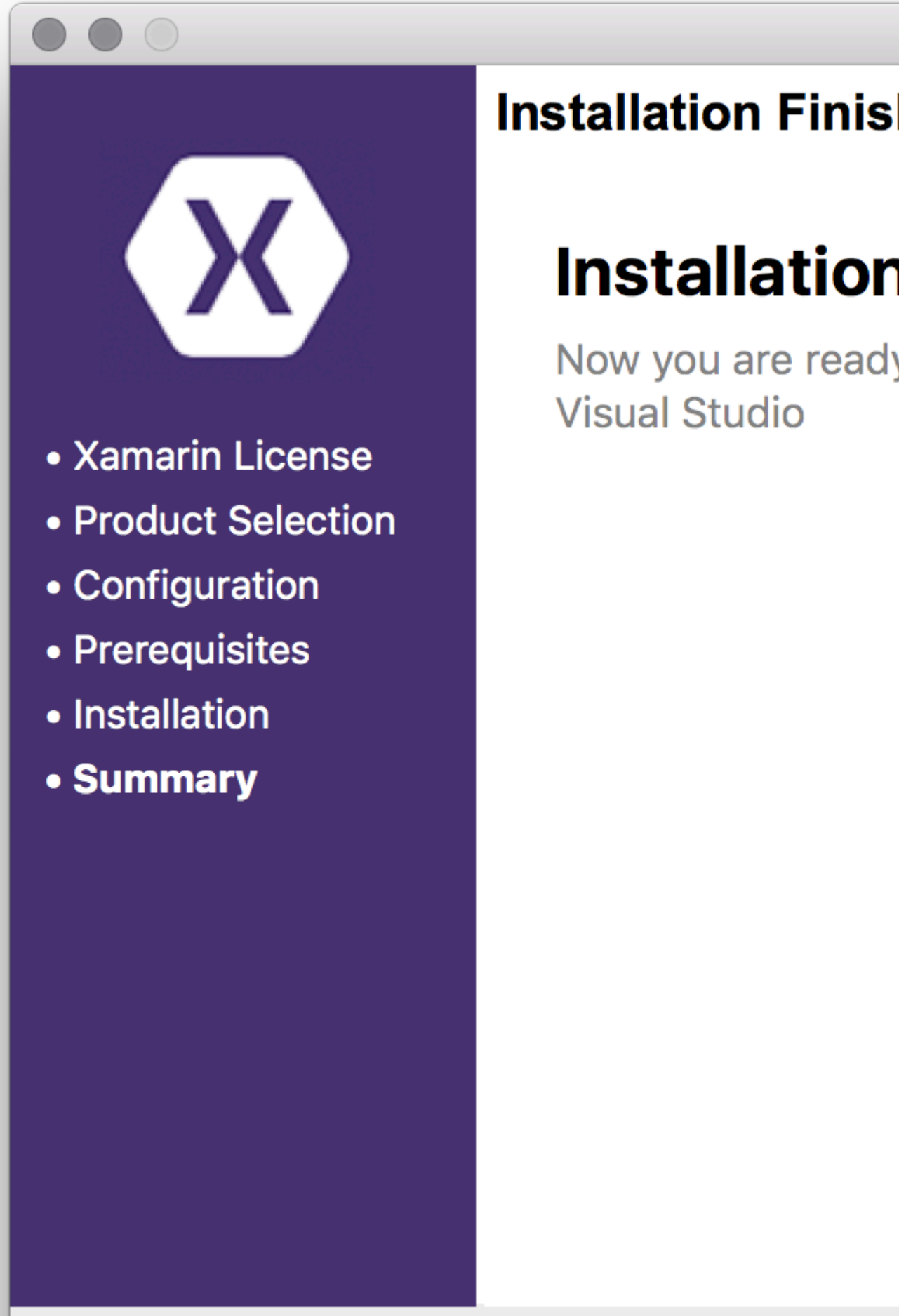
Чтобы начать процесс установки, вы должны прочитать и принять условия лицензии на программное обеспечение Xamarin. Установите флажок «Я согласен с условиями лицензии» и сделаю запись об автоматическом использовании и сообщении об ошибках.


Read and accept
Xamarin Software License
Last updated: November
1. ACCEPTANCE
PLEASE READ THIS AG
SOFTWARE (INCLUDING
IF YOU DO NOT AGREE
This Xamarin Software Li
a person) and Xamarin In
organization, or other enti
(ii) You represent and war
bind such entity to this Ag
such entity's behalf. For c
executed a subscription a
and conditions of this Agr
at https://store.xamarin.co
Agreement, electronically
2. DEFINITIONS
I accept the terms
Automatically send
Xamarin License
Prerequisites
Installation
Summary

будет автоматически загружать и выполнять каждый установщик. Если на последнем шаге был выбран Xamarin.Android, вам будет предложено выбрать место установки для Android SDK. Расположение по умолчанию является безопасным выбором в большинстве случаев, поэтому нажмите «Продолжить», чтобы продолжить.


Please configure
+
/Users/hankidesign/Li
Private copy of Androi
Size on disk: 338.
Download size: 948.
• Xamarin License
• Product Selection
• Configuration
• Prerequisites
• Installation
• Summary

Издание сообщества бесплатное и не требует входа в систему, но для использования функций Enterprise необходимо создать учетную запись и активировать пробную версию.


Installation Finis
Installation
Now you are ready
Visual Studio
• Xamarin License
• Product Selection
• Configuration
• Prerequisites
• Installation
• Summary

Hello World.

Hello World Application: Xamarin.Forms

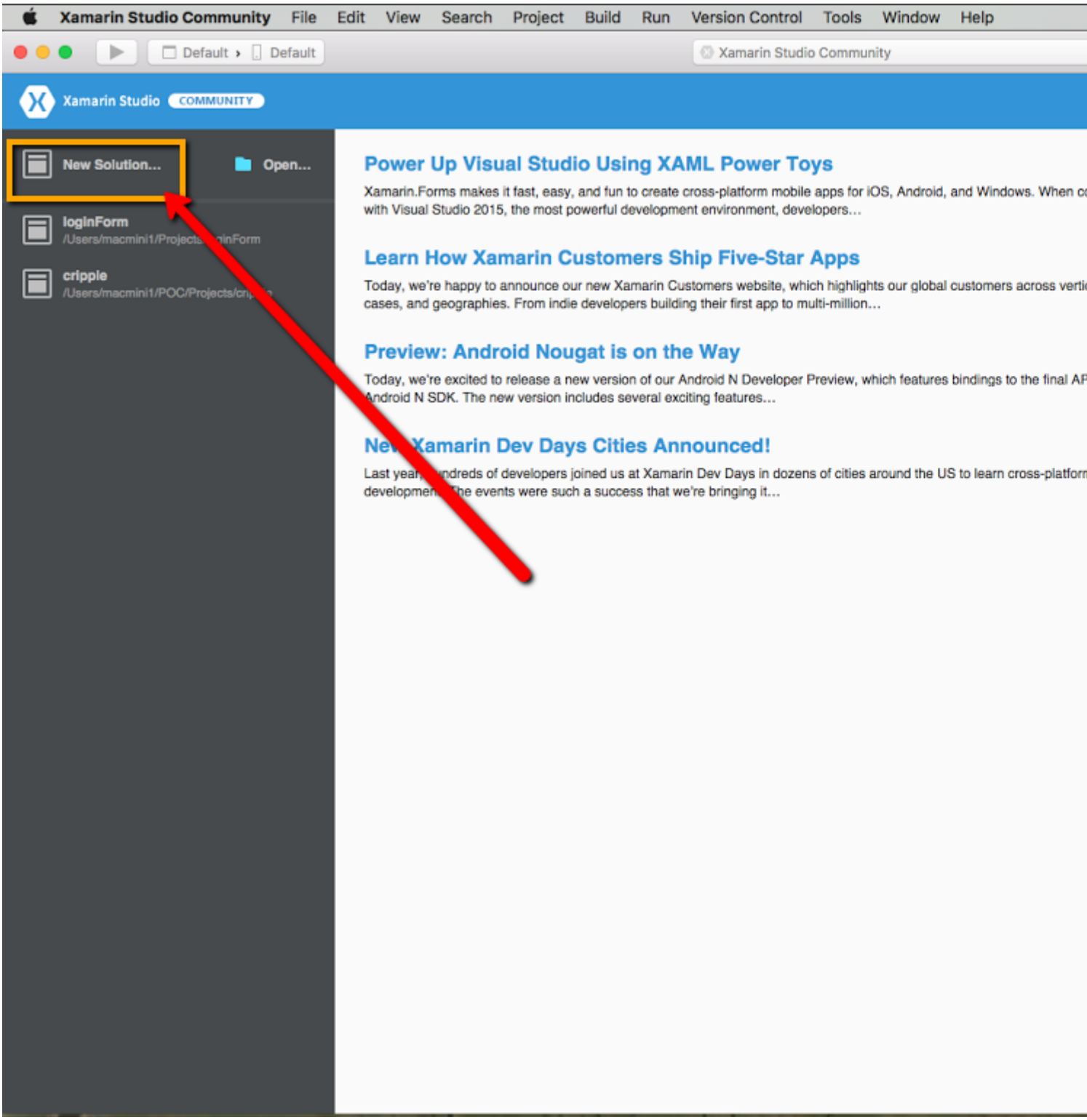
Что такое Xamarin Forms:

Xamarin.Forms - это новая библиотека, которая позволяет создавать собственные пользовательские интерфейсы для iOS, Android и Windows Phone с единой общей кодовой базы C #. Он предоставляет более 40 кросс-платформенных элементов управления и макетов, которые сопоставляются с собственными элементами управления во время выполнения, что означает, что ваши пользовательские интерфейсы полностью родны

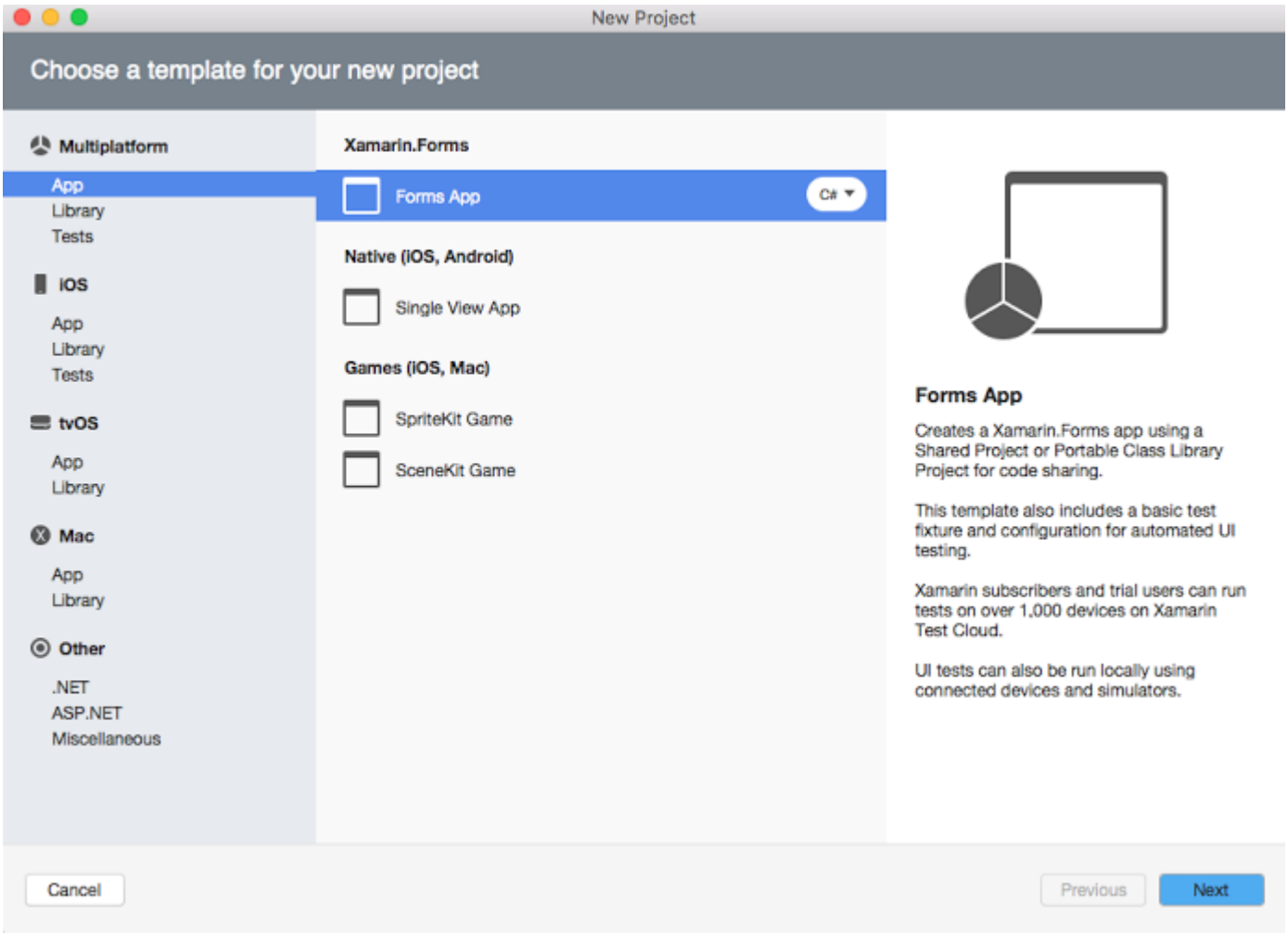
Шаг 1:

Создайте новое решение.

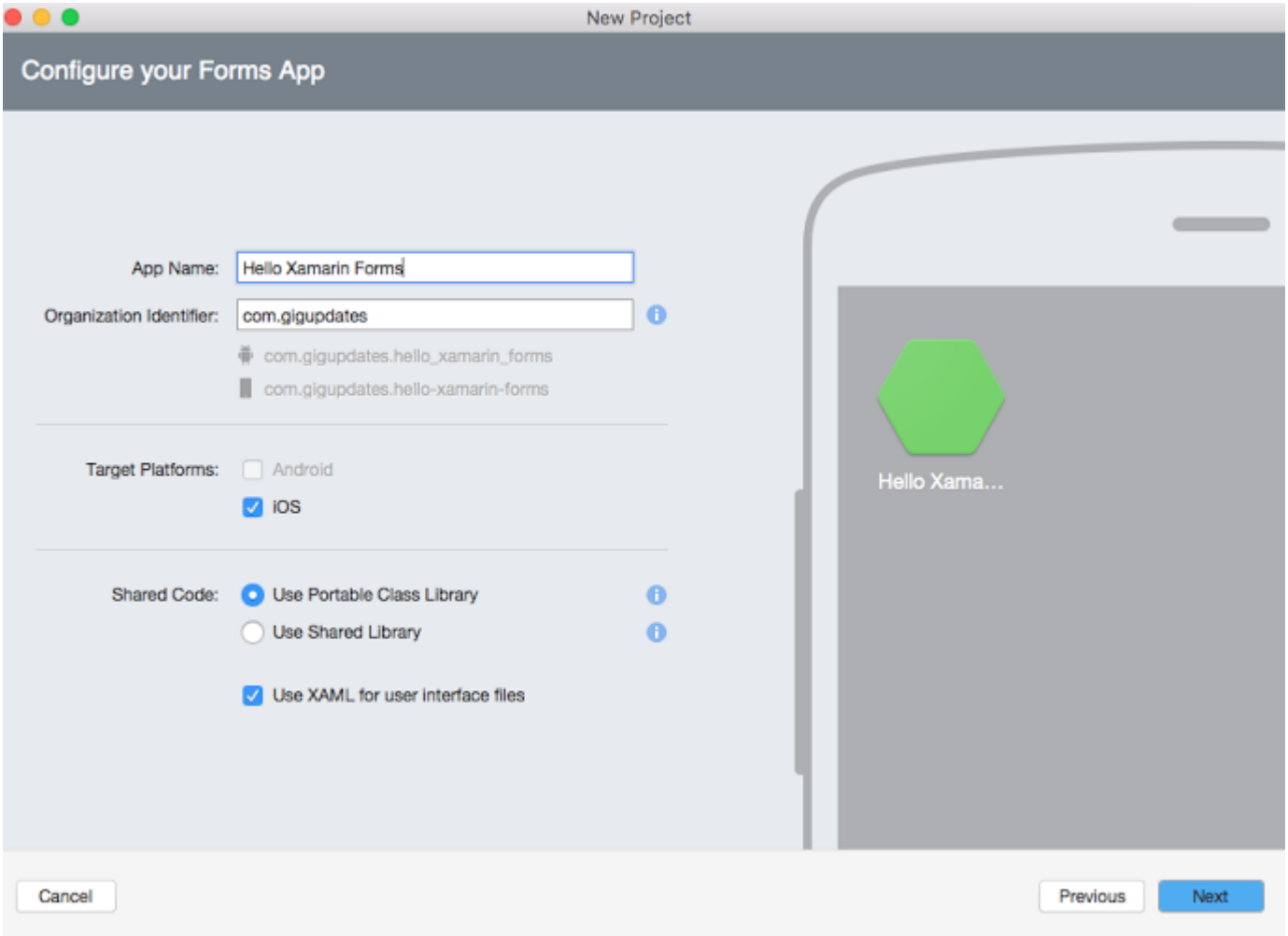
Нажмите «Новое решение»



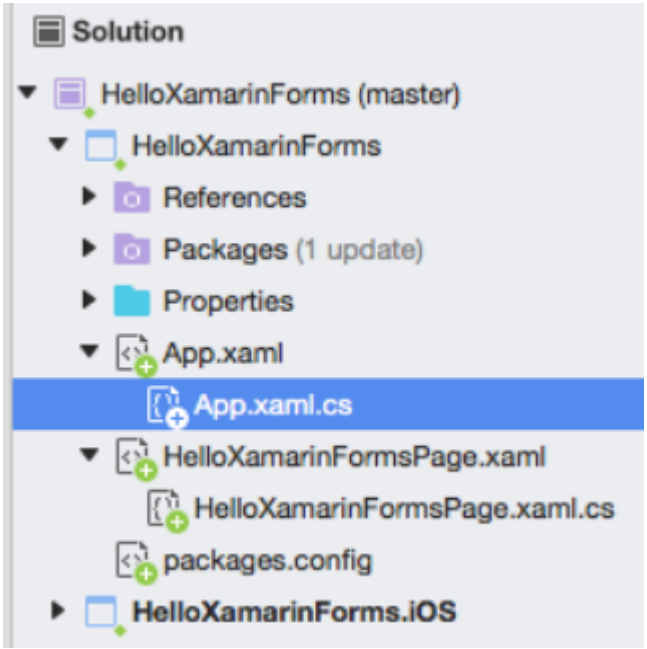
Шаг 2. Выберите приложение «Формы» и нажмите «Далее».



Шаг 3: добавьте имя приложения и нажмите «Далее».



Так будет выглядеть стрижка проекта при создании решения:



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();
        }
    }
}
```

Прочитайте Начало работы с xamarin онлайн: https://riptutorial.com/ru/xamarin/topic/899/начало-работы-с-xamarin

глава 2: Обмен кодами между проектами

Examples

Схема моста

Шаблон моста является одним из самых основных шаблонов проектирования инверсии управляющих элементов. Для Xamarin этот шаблон используется для ссылки на платформозависимый код из независимого от платформы контекста. Например: использование Android AlertDialog из Portable Class Library или Xamarin Forms. Ни один из этих контекстов не знает, что такое объект AlertDialog, поэтому вы должны обернуть его в поле для их использования.

```
// 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 implemenation
    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...
}
```

Шаблон локатора службы

Конструкция шаблона Service Locator представляет собой почти инъекцию зависимости. Подобно шаблону Bridge, этот шаблон можно использовать для ссылки на платформозависимый код из независимого от платформы контекста. Самое интересное, что этот шаблон основан на шаблоне одноэлементности - все, что вы помещаете в локатор

сервисов, будет деактивационным синглоном.

```
// 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
}
```

Прочитайте Обмен кодами между проектами онлайн:
https://riptutorial.com/ru/xamarin/topic/6183/обмен-кодами-между-проектами

глава 3: Проверка объектов по аннотациям

Вступление

mvc.net вводит аннотации данных для проверки модели. Это также можно сделать в Xamarin

Examples

Простой пример

Добавить пакет `System.ComponentModel.Annotations`

Определить класс:

```
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; }
}
```

Определите валидатор:

```
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
```

```
        );
    }
}
```

используйте валидатор:

```
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();
```

Полученный результат:

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

Прочитайте Проверка объектов по аннотациям онлайн:
https://riptutorial.com/ru/xamarin/topic/9720/проверка-объектов-по-аннотациям

кредиты

S. No	Главы	Contributors
1	Начало работы с xamarin	Akshay Kulkarni, Community, Gil Sand, hankide, Joel Martinez, Marius Ungureanu, Sven-Michael Stübe, thomasvdb
2	Обмен кодами между проектами	kellen lask, valdetero
3	Проверка объектов по аннотациям	Niek de Gooijer