

Разработка мобильных приложений для iOS и Android на C#

Андрей Басков, Touch Instinct



C#? Доклад проплачен?

- Да! (на самом деле нет 😞)
- Лямбда-выражения, делегаты, евенты
- LINQ
- Properties
- Generics
- Структуры
- dynamic
- В C# 5 async/await (mmm как сладко)

Асинхронный код раньше

```
MyApi.OnSomeMethod += () => {  
    InvokeOnMainThread( (result) => {  
        textView.Text = result;  
    });  
}
```

```
MyApi.SomeMethodAsync();
```

async/await

```
textView.Text = await MyApi.GetUrlAsync(  
    "http://touchin.ru");
```

C# vs Objective-C

Objective-C:

```
// ...  
    [button addTarget:self  
action:@selector(touchHandler:)  
forControlEvents:UIControlEventTouchUpInside];  
// ...  
  
-(void) touchHandler:(id)sender {  
    textView.text = @"some text";  
}
```

C# vs Objective-C

C#:

```
btn.TouchUpInside += (sender, e) => {  
    textView.Text = "Clicked!";  
};
```

C# vs Java

Java:

```
button.setOnClickListener(  
    new View.OnClickListener() {  
        public void onClick(View v) {  
            textView.setText("Clicked");  
        }  
    }  
);
```

C# vs Java

C#:

```
button.Click += (sender, e) {  
    textView.Text = "Clicked!";  
};
```


C# vs {0}

- Проще
- Чище
- Развивается очень быстро
- Меньше скобочек (но еще не Ruby/Python)
- Из Java песок сыплется
- [[[[После Objective-C] мир: квадратный] как:очень] и весь: вдвоеточиях];

Мир, дружба!

Mono

- Свободная реализация стандарта ECMA-334 (C#) и ECMA-335 (CLI)
- Развивается с 2004 года
- Основатель – Miguel de Icaza (GNOME, Ximian, Midnight Commander, WINE)
- Исходники, комьюнити, все как у людей
- На данный момент поддерживается C# 4.0 и большая часть BCL

Xamarin

- Ximian -> Novell -> Attachmate -> Xamarin
- Продает и саппортит Monotouch + Mono for Android
- Цены от 399\$ за лицензию
- На сайте отличные tutorиалы, документация, исходники типичных приложений
- Выпущено 4 книги
- <https://github.com/xamarin/monotouch-samples>

Ах, да, представиться 🤗

- 1.5 года опыта разработки с Monotouch
- Своя компания разрабатывающая мобильные приложения для iOS, Android, WP7
- Активно используем Monotouch, Mono for Android, Mono
- Наши приложения попадают в топы AppStore и Google Play
- Шарим код, пишем правильные архитектуры
- Нам уже год, 17 человек, Kinect, бинбегги, массажистки, блекджек

Что же такое Monotouch

- Mono framework с AOT компиляцией для ARM процессоров и Bindings к родным API
- Код пишется на C#
- UI используется родной, через C# обертки
- C#-зированный API (евенты, проперти, енумы)
- Среда разработки Monodevelop + XCode

Процесс компиляции

- Компилируется ваш код, библиотеки, VCL, обертки над нативными методами в IL
- Теоретически можно юзать любой язык (F#, IronPython, IronRuby, Nemerle, VB для особенных)
- IL преобразуется в машинный код с использованием AOT компиляции
- К коду приложения добавляется Mono Runtime с Garbage Collector'ом и всем остальным

AOT vs JIT

- Обычно в .Net и Mono машинный код генерируется в момент запуска – Just In Time компиляция
- В iOS нельзя компилировать код на лету, только статическая линковка
- Но мы заранее знаем архитектуру (ARM) поэтому можно компилировать код заранее – Ahead Of Time компиляция

Ограничения AOT

- Нету Emit, но Reflection остается
- Некоторые специфичные конструкции работать не будут, т.к. компилируются на лету
 - Generic Virtual Methods
 - P/Invokes in Generic Types
 - Некоторые LINQ expressions

Linking

- В момент компиляции в IL из VCL берется только тот код, который реально используется
- Таким же образом можно вырезать неиспользуемый код в своих либах
- Нужно для уменьшения размеров приложения

C#-изация API

- Подписка на события
- Установка свойств
- Привычные названия

```
var btn = new UIButton(new RectangleF(0, 0, 200, 80));  
btn.Enabled = true;  
btn.SetTitleColor(UIColor.FromRGB(255, 255, 0),  
UIControlState.Selected);  
btn.TouchUpInside += delegate {  
    // your code  
};  
window.Add(btn);
```

Обертки над нативными методами

Monotouch:

- Все сводится к P/Invoke метода `objc_msgSend` с нужными параметрами
- См код `Monotouch.ObjcRuntime.Messaging`

Mono for Android:

- Используется JNI (Java Native Interface)

Обертки над нативными методами

```
public virtual bool Enabled
{
    [Export("isEnabled")]
    get
    {
        // ...
        return Messaging.bool_objc_msgSend(base.Handle, UIControl.sellsEnabled);
    }
    [Export("setEnabled:")]
    set
    {
        // ...
        Messaging.void_objc_msgSend_bool(base.Handle,
            UIControl.selSetEnabled_, value);
    }
}
```

Структура приложения

- Практически соответствует такой же у нативного приложения
- AppDelegate, UIWindow, ViewControllers
- Для описания UI также используются nib файлы

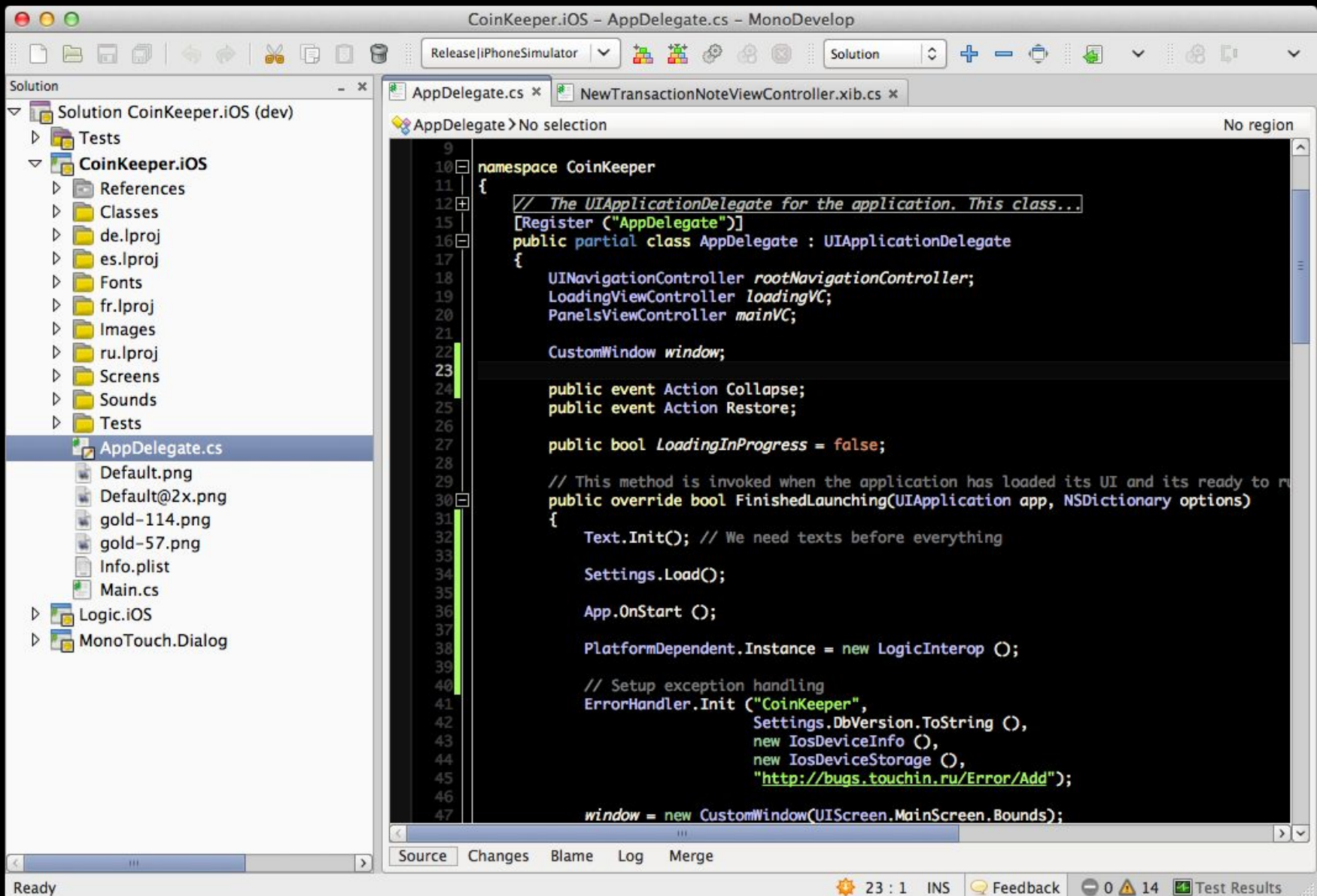
Дебаг

- Вместе с дебаг сборкой идет вся отладочная инфа + механизм удаленного дебага
- Раньше был по Wi-Fi сейчас по USB (быстрее)
- Дебаг полноценный с breakpoint'ами watch'ами итд

Garbage Collector

- Работает 😊
- Агрессивен – может часто вызываться
- Не забывайте – локальная переменная, значит обязательно соберется сборщиком
- Говорят что в играх, если много объектов может подтормаживать (актуально для Unity скорее)

Среда разработки - Monodevelop



Для UI - XCode

The screenshot displays the Xcode IDE interface for a project named "CoinKeeper.iOS.xcodeproj". The main window shows a storyboard for "NewTransactionViewController.xib" with a scroll view containing a segmented control with "Expense", "Income", and "Transfer" options, and a list of items: "California", "Brea", and "Burlingame". The code editor shows the following code:

```
// WARNING
// This file has been generated automatically by
// MonoDevelop to
// mirror C# types. Changes in this file made by drag-
// connecting
// from the UI designer will be synchronized back to C#,
// but
// more complex manual changes may not transfer correctly.

#import <UIKit/UIKit.h>
#import <Foundation/Foundation.h>
#import <CoreGraphics/CoreGraphics.h>

@interface NewTransactionViewController : UIViewController
{
    UITableView *_propertiesTbl;
    UIScrollView *_scrollView;
    UIView *_contentView;
    UIDatePicker *_datePicker;
    UIPickerView *_recurringPicker;
    UIScrollView *_fromScrollView;
    UIScrollView *_toScrollView;
    UISegmentedControl *_typeSelector;
    UILabel *_fromLbl;
    UILabel *_toLbl;
}

@property (nonatomic, retain) IBOutlet UITableView *
propertiesTbl;

@property (nonatomic, retain) IBOutlet UIScrollView *
scrollView;

@property (nonatomic, retain) IBOutlet UIView *contentView;

@property (nonatomic, retain) IBOutlet UIDatePicker *
datePicker;

@property (nonatomic, retain) IBOutlet UIPickerView *
recurringPicker;

@property (nonatomic, retain) IBOutlet UIScrollView *
fromScrollView;

@property (nonatomic, retain) IBOutlet UIScrollView *
toScrollView;

@property (nonatomic, retain) IBOutlet UISegmentedControl *
typeSelector;

@property (nonatomic, retain) IBOutlet UILabel *fromLbl;

@property (nonatomic, retain) IBOutlet UILabel *toLbl;

@end
```

The right sidebar shows the "Scroll View" property inspector with the following settings:

- Style: Default
- Scrollers: Shows Horizontal Scrollers, Shows Vertical Scrollers, Scrolling Enabled, Paging Enabled, Direction Lock Enabled
- Bounce: Bounces, Bounce Horizontally, Bounce Vertically
- Zoom: Min 1, Max 1
- Touch: Bounces Zoom, Delays Content Touches, Cancellable Content Touches
- View: Mode Scale To Fill, Tag 0, Interaction User Interaction Enabled, Multiple Touch, Alpha 1, Background Default
- Drawing: Opaque, Hidden, Clears Graphics Context, Clip Subviews, Autoresize Subviews

The bottom of the sidebar shows the "Objects" list with a "Label" selected, which has a description: "Label - A variably sized amount of static text." Below it, a "Round Rect Button" and a "Segmented Control" are also listed with their respective descriptions.

Интеграция со сторонними либами

- P/Invoke для C кода
- btouch для Objective-C Bindings
- В первый раз сложно, а потом халява
- Есть готовые обертки для популярных либ: Google Analytics, Flurry и др.

Минусы

- Размер (минимум 2-3Мб)
- Рантайм идет с каждым приложением
- Производительность (но не критично)
- Все равно придется учить UIKit/Android UI

Плюсики

- Код шаринг (но не забываем про ограниченный WP7)
- Не надо учить Objective-C (на самом деле нет)
- Проще разрабатывать (на самом деле быстрее)

А что на практике

- А на практике вполне себе хорошо
- CoinKeeper (iOS + Android + сервер) - удобная архитектура, сокращение кода, багов, времени разработки
- Наш супер [SecretProject] - тоже все пучком
- Omlet.ru – 2 недели на аппы под iPad и Android, код шаринг на уровне АПИ, быстро реагировали на изменение АПИ

Mono for Android

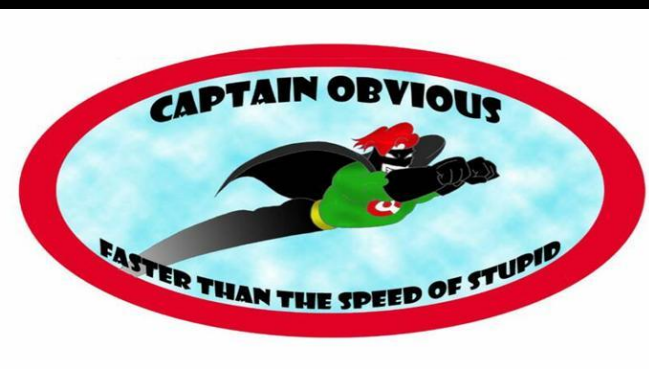
- Почти тоже самое, за исключением:
- JIT компиляция, можно EMIT (на андроиде вообще все можно, порно, смс вирусы, убийство младенцев)
- Одновременно живут две машины Mono VM и Dalvik VM
- GREFs
- Начиная с Android 1.6
- Свой редактор UI в Monodevelop (alpha)

Кроссплатформенность

- Шаринг в районе 30-60 процентов кода
- В основном работа с АПИ, с локальной базой, внутренняя бизнес логика
- UI все равно переписывается
- Xamarin.Mobile объединяющий основной мобильный функционал (фото, геолокация)
- Есть различные MVC фреймворки

Делайте хорошую архитектуру!

- Четкое разделение кода логики и UI
- Unit test'ы (на девайсе тоже – TouchUnit)
- Выносим все что можно, делаем общие методя для стандартных операций (Alert, InvokeOnMainThread итд)



Xobot OS

- Android переписанный на C#
- Быстрее обычного андроида из-за особенностей C#
- Просто концепт

Риски

Конкуренты

- Appcelerator
- PhoneGap
- Unity (тоже на Mono работает)

Вопросы!

И да, мы супер активно нанимаем!
Пишите!

Twitter: @AndreyBaskov

Email: ab@touchin.ru

Site: touchin.ru