Интенсив: "Правильный Xamarin для мобильных приложений"

**Спикер:** Вячеслав Черников, руководитель разработки, [Binwell](http://www.binwell.com/)

**Модератор:** Мария Горелкина, технологический евангелист, Microsoft

Содержание

[Часть 1. Что такое Xamarin.Forms 2](#_Toc465721932)

[Практическое задание: 2](#_Toc465721933)

[Дополнительно: 2](#_Toc465721934)

[Документация: 3](#_Toc465721935)

[Часть 2. Работаем с REST 4](#_Toc465721936)

[Практическое задание: 4](#_Toc465721937)

[Дополнительно: 5](#_Toc465721938)

[Документация: 5](#_Toc465721939)

[Часть 3. Архитектура приложения на Xamarin.Forms 6](#_Toc465721940)

[Практическое задание: 6](#_Toc465721941)

[Документация: 6](#_Toc465721942)

[Дополнительные ресурсы 7](#_Toc465721943)

[Материалы лабораторных работ: 7](#_Toc465721944)

[Обучающие ресурсы: 7](#_Toc465721945)

[Сообщество: 7](#_Toc465721946)

[Участие в технологической акселерации Microsoft: 7](#_Toc465721947)

# Часть 1. Что такое Xamarin.Forms

Знакомимся с Xamarin и cоздаем первое приложение на Xamarin.Forms

## Практическое задание:

1. Создайте свое первое приложение на Xamarin.Forms, используя установленное ПО на вашем компьютере.

1. Добавьте необходимые элементы интерфейса, например, кнопку, текстовое поле, список и т.д.

1. Используйте знания, полученные в ходе теоретической части и внесите изменения в код для повышения производительности вашего приложения:

**Предварительная компиляция Xaml**

Включаем компиляцию отдельной XAML-страницы (файл MainPage.xaml.cs, например)

**using** Xamarin.Forms.Xaml;

...

[XamlCompilation (XamlCompilationOptions.Compile)]

**public** **class** **MainPage** : **ContentPage**

{

...

}  
  
Активируем компиляцию по всему проекту — добавляем новую строку в конец файла Properties/AssemblyInfo.cs в PCL-проекте:

...

[assembly: XamlCompilation(XamlCompilationOptions.Compile)]

Ссылка на документацию: [https://developer.xamarin.com/guides/xamarin-forms/deployment,\_testing,\_and\_metrics/performance/#xamlc](https://developer.xamarin.com/guides/xamarin-forms/web-services/#xamlc)

**Повторное использование созданных ячеек**

В последних версиях Xamarin.Forms реализован механизм повторного использования созданных ячеек и для его активации необходимо задать свойству CachingStrategy значение ListViewCachingStrategy.RecycleElement.  
  
Реализация в XAML:

<ListView CachingStrategy="RecycleElement">

...

</ListView>

Реализация на C#:

**var** listView = **new** ListView(ListViewCachingStrategy.RecycleElement);

После этого созданные экземпляры ViewCell начинают использоваться повторно и к ним просто подключаются (через Binding) необходимые модели данных по мере отображения.

Ссылка на документацию: <https://blog.xamarin.com/optimizing-xamarin-forms-apps-for-maximum-performance/>

## **Дополнительно:**

**Изменения для оптимизации загрузки изображений в списке**

Воспользуйтесь компонентом FFImageLoading:  
[https://github.com/luberda-molinet/FFImageLoading](https://www.visualstudio.com/dev-essentials/)

Компонент доступен в Nuget и позволяет решить сразу несколько задач:

* фоновая загрузка изображений с возможностью повтора запросов;
* использование placeholder во время загрузки;
* автоматическое масштабирование изображений до размеров контрола, включая удаление Alpha-канала на Android для еще большей производительности;
* возможность применения трансформаций к изображениям после загрузки (обрезать до круга или другой формы, наложить blur или другой спец. эффект);
* fade-анимация появления изображения после загрузки.

Вот так использовать компонент при разработке на 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"

x:Class="Sample.Pages.MainPage"

xmlns:ffimageloading="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"

Title="FFImageLoading Demo">

<ContentPage.Content>

<ffimageloading:CachedImage HorizontalOptions="Center" VerticalOptions="Center"

WidthRequest="300" HeightRequest="300"

DownsampleToViewSize="true"

Source = "https://unsplash.it/600/600/?random"

LoadingPlaceholder = "placeholder.png">

</ffimageloading:CachedImage>

</ContentPage.Content>

</ContentPage>

Вот так можно еще немного улучшить работу FFImageLoading (класс AppDelegate.cs для iOS и MainActivity.cs для Android):

**var** config = **new** Configuration

{

HttpClient = **new** HttpClient(**new** ModernHttpClient.NativeMessageHandler()), *//используем быстрые библиотеки для загрузки*

FadeAnimationDuration = 250, *//ускоряем анимацию появления*

};

FFImageLoading.ImageService.Instance.Initialize(config);

## Документация:

Основные рекомендации по улучшению производительности:

* Xamarin.Forms: [https://developer.xamarin.com/guides/xamarin-forms/deployment,\_testing,\_and\_metrics/performance/](https://github.com/JamesNK/Newtonsoft.Json)
* Non-platfrom specific: [https://developer.xamarin.com/guides/cross-platform/deployment%2C\_testing%2C\_and\_metrics/memory\_perf\_best\_practices](https://github.com/luberda-molinet/FFImageLoading)

# Часть 2. Работаем с REST

 Подключаем сервер и моки

## **Практическое задание:**

1. Создайте REST сервис, исходя из вашей предметной области, используя ресурс <https://www.mockable.io>

Или воспользуйтесь стандартным примером: [http://demo4404797.mockable.io/speakers](https://developer.xamarin.com/guides/cross-platform/deployment,_testing,_and_metrics/memory_perf_best_practices)

1. Воспользуйтесь, знаниями полученными в ходе теоретической части для подключения приложения к внешнему REST Сервису:

**Библиотека Refit**

Refit позволяет описать спецификации для работы с REST-сервисом в виде простого Interface с понятным набором входных и выходных параметров, включая возможность манипулировать HTTP-заголовками для отдельных запросов.

[https://github.com/paulcbetts/refit](https://channel9.msdn.com/Shows/XamarinShow)

Здесь мы для примера возьмем демо-API сервиса [httpbin.org](mailto:v-anaso@microsoft.com):

[Headers("Accept: application/json")]

public interface IHttpbinApi

{

[Get("/basic-auth/{username}/{password}")]

Task<AuthResult> BasicAuth(string username, string password, [Header("Authorization")] string authToken, CancellationToken ctx);

[Get("/cache")]

Task<HttpResponseMessage> CheckIfModified([Header("If-Modified-Since")] string lastUpdateAtString, CancellationToken ctx);

[Post("/post")]

Task<HttpResponseMessage> FormPost([Body(BodySerializationMethod.UrlEncoded)] FormData data, CancellationToken ctx);

}

После описания данного интерфейса, он подается на вход для Refit:

var client = new HttpClient(new NativeMessageHandler())

{

BaseAddress = new Uri("http://httpbin.org")

};

\_httpbinApiService = RestService.For<IHttpbinApi>(client);

**Json.NET и DTO**

<https://github.com/JamesNK/Newtonsoft.Json>

Сами данные можно при необходимости можно расширить аттрибутами из библиотеки [Json.net](https://github.com/nishanil/Dev-Days-HOL/tree/master/03%20DevOps-Labs), так как именно она используется в Refit:

public class AuthResult

{

[JsonProperty("authenticated")]

public bool IsAuthenticated { get; set; }

[JsonProperty("user")]

public string Login { get; set; }

}

В Refit в качестве выходного значения можно получить уже преобразованные объекты [DTO](https://telegram.me/xamarin_russia) или HttpResponseMessage. Последний позволяет получить информацию о запросе и ответе, что может быть полезно при отладке. При желании также может использоваться ModernHttpClient при создании HttpClient.

## Дополнительно:

**DAL DataServices**

public static void Init(bool isMock = false) {

if (isMock)

DependencyService.Register<ISessionDataService, Mock.SessionDataService>();

else

DependencyService.Register<ISessionDataService, Remote.SessionDataService>();

}

public static ISessionDataService Session => DependencyService.Get<ISessionDataService>();

 Ссылка на документацию: [https://developer.xamarin.com/guides/xamarin-forms/dependency-service/introduction/](https://blog.xamarin.com/introducing-the-azure-track-in-xamarin-university/)

## Документация:

* Overview

[https://developer.xamarin.com/guides/xamarin-forms/web-services/](https://developer.xamarin.com/guides/xamarin-forms/dependency-service/introduction/)

* Consuming Web-services

<https://developer.xamarin.com/guides/xamarin-forms/web-services/consuming/rest/>

* Authenticating Access to Web Services

[https://developer.xamarin.com/guides/xamarin-forms/web-services/authentication/rest/](https://github.com/paulcbetts/refit)

# Часть 3. Архитектура приложения на Xamarin.Forms

Приводим структуру проекта в порядок

## Практическое задание:

1. Преобразите структуру приложения в соответствии с MVVM - шаблоном, где:

* ViewModel – управление состоянием View, контекст для биндинга (свойства и команды), логика переходов между экранами,
* Model – получение информации из базового объекта

1. Воспользуйтесь знаниями, полученными в ходе теоретической части и примером приложения Xamarin Evolve для усовершенствования архитектуры вашего приложения

**Пример**

* Приложение Xamarin Evolve: [https://github.com/xamarinhq/app-evolve](https://developer.xamarin.com/guides/xamarin-forms/deployment,_testing,_and_metrics/performance/)

## Документация:

<https://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_2_-_architecture/>

# Дополнительные ресурсы

## Материалы лабораторных работ:

Создание простейшего приложения на Xamarin Android/iOS:

[https://github.com/nishanil/Dev-Days-HOL/tree/master/00%20Quickstart/Phoneword](https://mva.microsoft.com/en-US/training-courses-embed/xamarin-for-absolute-beginners-16182)

Создание приложения «WeatherApp» на Xamarin Android/iOS:

<https://github.com/nishanil/Dev-Days-HOL/tree/master/00%20Quickstart/WeatherApp>

Создание приложений на Xamarin.Forms: <https://github.com/nishanil/Dev-Days-HOL>

* [Создание приложения](http://httpbin.org/)
* [Использование облачного бэкэнда в Azure](https://github.com/nishanil/Dev-Days-HOL/tree/master/02%20Cloud-Labs)
* [Мобильный DevOps для Xamarin приложений](https://developer.xamarin.com/guides/xamarin-forms/web-services/authentication/rest/)

## Обучающие ресурсы:

**Все обучающие ресурсы на одной странице, добавьте в закладки:** <http://apptractor.ru/xamarin/>

Xamarin University – 10 бесплатных курсов при регистрации:

<https://university.xamarin.com/>

Работа с облаком Azure в Xamarin приложении:

[https://blog.xamarin.com/introducing-the-azure-track-in-xamarin-university/](https://github.com/nishanil/Dev-Days-HOL/tree/master/00%20Quickstart/Phoneword)

MVA: [https://mva.microsoft.com/en-US/training-courses-embed/xamarin-for-absolute-beginners-16182](mailto:magore@microsoft.com)

eDx: <https://www.edx.org/course/introduction-xamarin-forms-microsoft-dev215x>

Pluralsight на 3 месяца при регистрации в программе Dev Essnetials:

[https://www.visualstudio.com/dev-essentials/](https://github.com/nishanil/Dev-Days-HOL/tree/master/01%20Dev-Labs)

Колонка Xamarin в блоге Microsoft:

<https://habrahabr.ru/search/?target_type=posts&q=%5Bxamarincolumn%5D&order_by=date>

Xamarin Show на Channel9: [https://channel9.msdn.com/Shows/XamarinShow](https://ru.wikipedia.org/wiki/DTO)

## Сообщество:

**Встречи сообщества**: [https://www.meetup.com/Moscow-Xamarin-Meetup/](https://github.com/xamarinhq/app-evolve)

Канал в Telegram: [https://telegram.me/xamarin\_russia](http://demo4404797.mockable.io/speakers)

Группа ВК: <https://vk.com/xamarin>

## Участие в технологической акселерации Microsoft:

**Напишите нам** [magore@microsoft.com](https://www.meetup.com/Moscow-Xamarin-Meetup/) , [v-anaso@microsoft.com](https://developer.xamarin.com/guides/xamarin-forms/deployment,_testing,_and_metrics/performance/)