**Отчет по Главе 1: Введение в .NET MAUI**

**1. Введение**

Данная глава посвящена введению в технологию .NET MAUI (Multi-platform App UI). Значимость этой темы обусловлена тем, что .NET MAUI является современной и важной технологией для разработки кроссплатформенных приложений, включая настольные. Она позволяет создавать приложения, работающие на Windows, macOS, Android и iOS, используя единый код на C# и XAML, что существенно упрощает процесс разработки и поддержания программных продуктов. Изучение данной главы является первым шагом в освоении .NET MAUI в рамках учебной практики.

**2. Основная часть**

**2.1. Основные концепции и принципы**

.NET MAUI (Multi-platform App UI) – это кроссплатформенный фреймворк для создания нативных мобильных и настольных приложений с использованием .NET и C#. Он является развитием Xamarin.Forms и позволяет разработчикам использовать единую кодовую базу и единый проект для таргетирования множества платформ.

Ключевые концепции и принципы .NET MAUI:

* **Единый проект:** Вся разработка ведется в рамках одного проекта, который содержит общий код и ресурсы.
* **Единая кодовая база:** Логика приложения и пользовательский интерфейс (UI) пишутся преимущественно один раз на C# и XAML.
* **Нативные пользовательские интерфейсы:** .NET MAUI транслирует общий UI, определенный в XAML или C#, в нативные элементы управления для каждой целевой платформы, обеспечивая привычный для пользователя опыт.
* **Доступ к нативным API:** Предоставляет абстракции для общих платформенных функций (Essentials) и возможность прямого вызова специфичных для платформы API.
* **Hot Reload:** Поддержка "горячей перезагрузки" для XAML и C#, что ускоряет итерации разработки UI.

**2.2. Описание процесса установки и настройки необходимых инструментов**

Для начала разработки приложений на .NET MAUI необходимо настроить среду разработки. Основным инструментом является Visual Studio (рекомендуется версия 2022 или новее).

Процесс установки:

1. **Запуск Visual Studio Installer:** Это приложение используется для управления установками Visual Studio.
2. **Изменение существующей установки или установка новой:** Если Visual Studio уже установлен, необходимо нажать "Изменить". В противном случае – выбрать новую установку.
3. **Выбор рабочей нагрузки:** На вкладке "Рабочие нагрузки" необходимо выбрать компонент ".NET Multi-platform App UI development". Эта нагрузка включает все необходимые SDK, инструменты и шаблоны проектов для .NET MAUI.
4. **Установка:** Подтвердить выбор и дождаться завершения загрузки и установки всех компонентов.

**2.3. Примеры кода с пояснениями**

После установки создается первое приложение .NET MAUI с использованием стандартного шаблона. Рассмотрим ключевые файлы и их содержимое.

* **MauiProgram.cs**: Точка входа приложения. Здесь конфигурируется и создается хост приложения.

// Файл: MauiProgram.cs

using Microsoft.Extensions.Logging;

namespace HelloMauiApp;

public static class MauiProgram

{

public static MauiApp CreateMauiApp()

{

var builder = MauiApp.CreateBuilder();

builder

.UseMauiApp<App>() // Указывает класс приложения App

.ConfigureFonts(fonts => // Конфигурация шрифтов

{

fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");

fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");

});

#if DEBUG

builder.Logging.AddDebug(); // Включение логирования в режиме отладки

#endif

return builder.Build(); // Сборка и возврат экземпляра MauiApp

}

}

*Пояснение:* В этом файле создается MauiAppBuilder, регистрируется класс приложения App, добавляются шрифты и настраивается логирование.

* **App.xaml.cs**: Code-behind файл для App.xaml, определяет класс приложения.

// Файл: App.xaml.cs

namespace HelloMauiApp;

public partial class App : Application

{

public App()

{

InitializeComponent(); // Инициализация компонентов, определенных в App.xaml

MainPage = new AppShell(); // Установка основной страницы приложения

}

}

*Пояснение:* InitializeComponent() загружает XAML-ресурсы, определенные в App.xaml. MainPage устанавливает стартовый UI приложения, часто это AppShell для навигации или непосредственно страница.

* **MainPage.xaml**: XAML-разметка для основной страницы.

<!-- Файл: MainPage.xaml -->

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="HelloMauiApp.MainPage"

Title="Home">

<ScrollView>

<VerticalStackLayout

Spacing="25"

Padding="30,0"

VerticalOptions="Center">

<Image

Source="dotnet\_bot.png"

SemanticProperties.Description="Cute dot net bot waving hi to you!"

HeightRequest="185"

HorizontalOptions="Center" />

<Label

Text="Hello, World!"

SemanticProperties.HeadingLevel="Level1"

FontSize="32"

HorizontalOptions="Center" />

<Button

x:Name="CounterBtn"

Text="Click me"

SemanticProperties.Hint="Counts the number of times you click"

Clicked="OnCounterClicked"

HorizontalOptions="Center" />

</VerticalStackLayout>

</ScrollView>

</ContentPage>

*Пояснение:* Определяет ContentPage с изображением, меткой и кнопкой, размещенными в VerticalStackLayout. Атрибут x:Class связывает этот XAML с классом HelloMauiApp.MainPage.

* **MainPage.xaml.cs**: Логика для MainPage.xaml.

// Файл: MainPage.xaml.cs

namespace HelloMauiApp;

public partial class MainPage : ContentPage

{

int count = 0;

public MainPage()

{

InitializeComponent(); // Загрузка XAML и создание UI-объектов

}

private void OnCounterClicked(object sender, EventArgs e)

{

count++;

CounterBtn.Text = $"Clicked {count} times"; // Обновление текста кнопки

SemanticScreenReader.Announce(CounterBtn.Text); // Для средств чтения с экрана

}

}

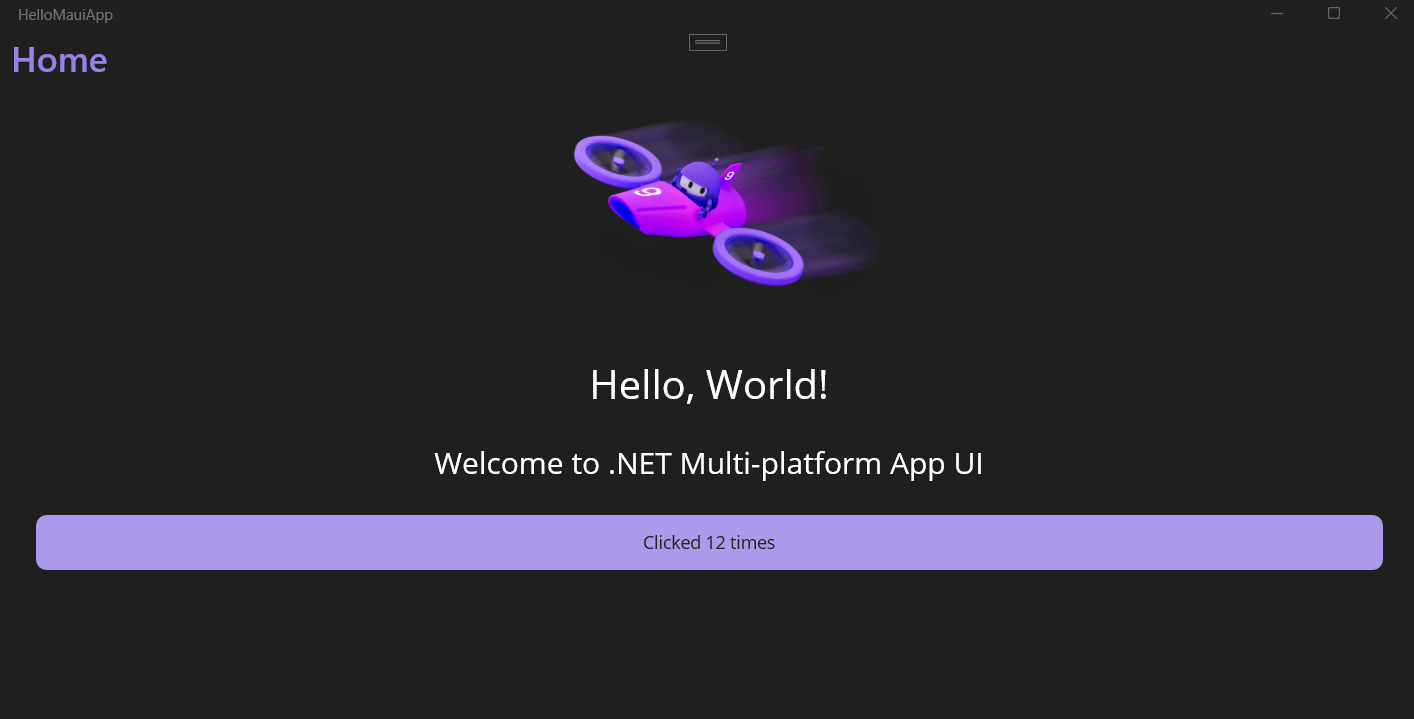
*Пояснение:* InitializeComponent() обязателен. Метод OnCounterClicked является обработчиком события нажатия кнопки CounterBtn (имя которой задано через x:Name в XAML).

**3. Заключение**

Изучение основных концепций .NET MAUI, процесса установки среды разработки и структуры первого приложения является фундаментальным. Эти знания позволяют понять, как .NET MAUI обеспечивает разработку кроссплатформенных приложений, включая настольные, с единой кодовой базой. Понимание принципов работы MauiProgram.cs, App.xaml.cs и взаимодействия XAML со связанным кодом C# закладывает основу для дальнейшего, более глубокого освоения фреймворка и его применения в разработке функциональных настольных приложений.

**4. Скриншоты и примеры**

Для демонстрации работы стандартного шаблона приложения .NET MAUI был создан проект HelloMauiApp и запущен на платформе Windows.

  
*Рис. 1.1. Работа стандартного приложения .NET MAUI HelloMauiApp на платформе Windows."*

**Процесс выполнения примера:**

1. Создание нового проекта .NET MAUI в Visual Studio 2022 с использованием стандартного шаблона.
2. Выбор целевой платформы "Windows Machine" для запуска.
3. Запуск приложения без внесения изменений в исходный код шаблона.  
   Приложение успешно скомпилировалось и запустилось, отобразив интерфейс, как показано на скриншоте. Кнопка "Click me" корректно реагировала на нажатия, изменяя свой текст и увеличивая счетчик. Проблем в процессе выполнения стандартного примера не возникло.

**5. Документация**

* **Ссылка на GitHub с исходным кодом:** https://github.com/RaniZee/.NET-MAUI-Showcase-Cross-Platform-Applications
* **Описание выполненных шагов:**
  1. Изучены материалы по теме "Введение в .NET MAUI" (глава 1) с ресурса [metanit.com/sharp/maui/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fmetanit.com%2Fsharp%2Fmaui%2F).
  2. Проверена и подтверждена корректность установки Visual Studio 2022 и рабочей нагрузки ".NET Multi-platform App UI development".
  3. Создан новый проект .NET MAUI с именем HelloMauiApp с использованием стандартного шаблона.
  4. Проанализирована структура созданного проекта, включая файлы MauiProgram.cs, App.xaml.cs, AppShell.xaml (если используется), MainPage.xaml и MainPage.xaml.cs.
  5. Приложение успешно скомпилировано и запущено на локальной машине Windows.
  6. Освоены основные концепции работы .NET MAUI и принцип единого проекта.