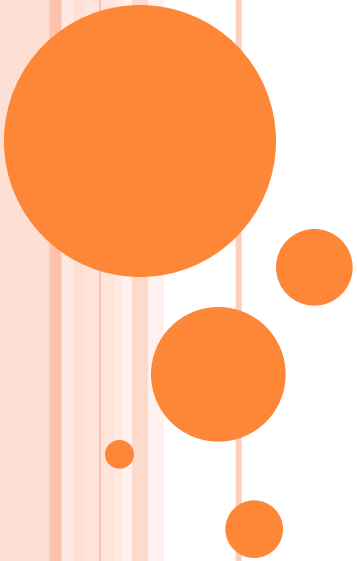


ХРАНЕНИЕ ДАННЫХ. ФАЙЛЫ. XAMARIN FORMS

Махнев А.А.

ЧЕТЫРЕ ТИПА ХРАНИЛИЩА ДАННЫХ

1. Коллекция Properties в классе Application, которая предназначена для хранения каких-то временных данных

2. Настройки (Preferences) операционной системы для хранения каких-то атомарных данных типа int или string

3. Файлы в файловой системе

4. Базы данных

Свойство Properties класса Application

- Представляет словарь, где каждому ключу с типом string сопоставляется некоторое значение.
- Содержимое этого словаря автоматически сохраняется перед тем, как приложение завершается, поэтому при новом запуске приложения мы сможем использовать ранее сохраненные в словаре данные.

СВОЙСТВО PROPERTIES КЛАССА APPLICATION

- При этом мы не ограничены только классом Application, в котором собственно определено данное свойство. Мы также можем к нему обращаться в коде страниц с помощью выражения **App.Current.Properties**.

- Принципы работы с Properties те же, что и со стандартными словарями

ДОБАВЛЕНИЕ ДАННЫХ В СЛОВАРЬ

- App.Current.Properties.Add("name", "Tom");
- // или так
- App.Current.Properties["name"] = "Tom";

ПОЛУЧЕНИЕ ЗНАЧЕНИЯ

- string name = App.Current.Properties["name"];

ПОЛУЧЕНИЕ С ПРОВЕРКОЙ НА НАЛИЧИЕ

```
object name = "";
if(App.Current.Properties.TryGetValue("name", out name))
{
    // выполняем действия, если в словаре есть ключ "name"
}
```

УДАЛЕНИЕ

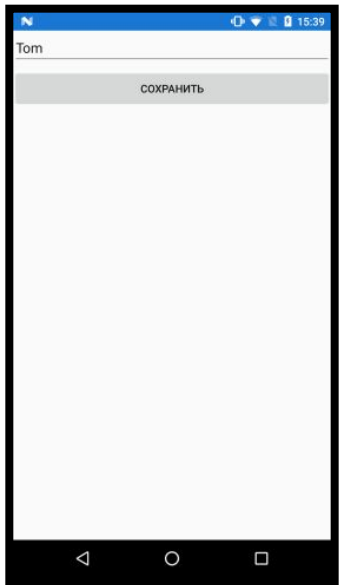
- App.Current.Properties.Remove("name");

ПРИМЕР ПРОЕКТА СТРАНИЦЫ MAINPAGE С ТЕКСТОВЫМ ПОЛЕМ И КНОПКОЙ

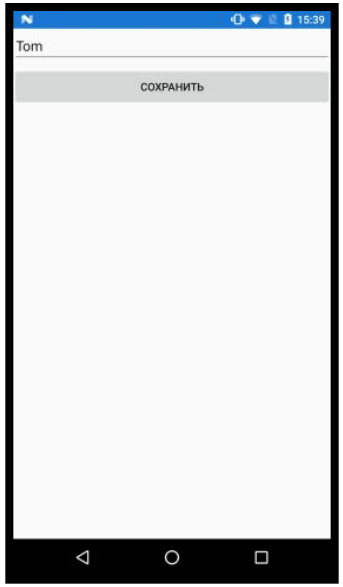
```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="HelloApp.MainPage">
  <StackLayout>
    <Entry x:Name="nameBox" />
    <Button Text="Сохранить" Clicked="OnClick" />
  </StackLayout>
</ContentPage>
```

ДОБАВЛЕНИЕ И ИЗВЛЕЧЕНИЕ ИЗ СЛОВАРЯ PROPERTIES ВВЕДЕННОГО В ТЕКСТОВОЕ ПОЛЕ ТЕКСТА

```
protected override void OnAppearing()
        {
            object name = "";
            if (App.Current.Properties.TryGetValue("name", out name))
            {
                nameBox.Text = (string)name;
            }
            base.OnAppearing();
        }
```



ДОБАВЛЕНИЕ И ИЗВЛЕЧЕНИЕ ИЗ СЛОВАРЯ PROPERTIES ВВЕДЕННОГО В ТЕКСТОВОЕ ПОЛЕ ТЕКСТА




```
        private void OnClick(object sender, EventArgs e)
        {
            string value = nameBox.Text;

            App.Current.Properties["name"] = value;
        }
```

НАСТРОЙКИ ПРИЛОЖЕНИЯ

- Каждая система предполагает механизм настроек пользователя, которые можно сохранять для последующего использования.
- Например, на Android это объект Preference, в iOS это объект CFPreferences, а в Windows 10 настройки доступны через объект ApplicationData.
- В Xamarin Forms для работы с настройками мы можем использовать функциональность из пакета **Xamarin Essentials**, который добавляется в проект Xamarin Forms по умолчанию.

ВСЕ НАСТРОЙКИ ДОСТУПНЫ ЧЕРЕЗ СВОЙСТВО **PREFERENCES**

- Методы:

- **void Set(key, value)**: сохранение значения value по ключу key

- **string Get(key, defaultValue)**: получение значения по ключу key. Если же настроек с таким ключом нет, то возвращается значение по умолчанию - defaultValue

ВСЕ НАСТРОЙКИ ДОСТУПНЫ ЧЕРЕЗ СВОЙСТВО **PREFERENCES**

- Методы:

- **void Remove(key)**: удаляет настройку с ключом key

- **void Clear()**: удаляет все настройки

- **bool ContainsKey(key)**: возвращает true, если имеется настройка с ключом key

ПРИМЕР СОХРАНЕНИЯ ПО КЛЮЧУ

- // сохраняем значение Tom по ключу name
- Preferences.Set("name", "Tom");

ПОЛУЧЕНИЕ СОХРАНЕННОГО ЗНАЧЕНИЯ ПО КЛЮЧУ

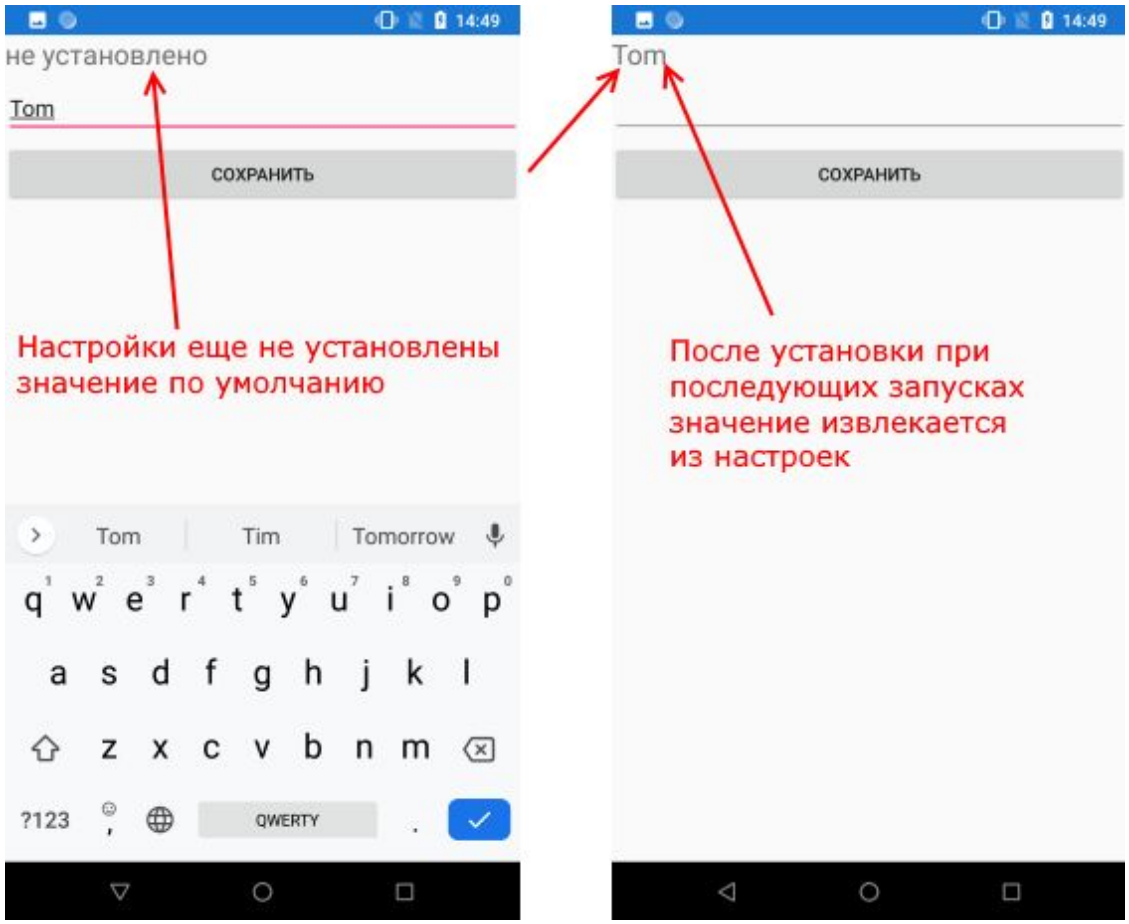
- string name = Preferences.Get("name");
- // если объекта нет используем значение по умолчанию - "не известно"
- string name2 = Preferences.Get("name", "не известно");

УДАЛЕНИЕ

- Preferences.Remove("name");

ПРИМЕР

- Определим на странице **текстовое поле** для ввода, **кнопку** для сохранения введенного значения в настройках и **метку** для вывода из настроек



КОД 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="HelloApp.MainPage">
  <StackLayout>
    <Label x:Name="nameLabel" />
    <Entry x:Name="nameBox" />
    <Button Text="Сохранить"
Clicked="OnClick" />
  </StackLayout>
</ContentPage>
```

ПОЛУЧЕНИЕ И СОХРАНЕНИЕ ДАННЫХ В НАСТРОЙКАХ

```
protected override void OnAppearing()
        {
            string name = Preferences.Get("name", "не установлено");
            nameLabel.Text = name;
            base.OnAppearing();
        }
```

ПОЛУЧЕНИЕ И СОХРАНЕНИЕ ДАННЫХ В НАСТРОЙКАХ

```
        private void OnClick(object sender, EventArgs e)
        {
            string value = nameBox.Text;
            nameLabel.Text = value;
            Preferences.Set("name", value);
        }
```

РАБОТА С ФАЙЛАМИ

- Каждая мобильная платформа имеет свою специфику, свою файловую систему, свою структуру каталогов.

- Работа с файлами и каталогами в Xamarin Forms производится также, как и в целом .NET - с помощью классов из пространства имен **System.IO**

ПРИМЕР ПРОЕКТА ДЛЯ РАБОТЫ С ФАЙЛАМИ

```
<?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="HelloApp.MainPage">
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="*" />
      <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <StackLayout Orientation="Horizontal">
      <Entry x:Name="fileNameEntry"
HorizontalOptions="FillAndExpand" />
      <Button Text="Сохранить" Clicked="Save" />
    </StackLayout>

```

ПРИМЕР ПРОЕКТА ДЛЯ РАБОТЫ С ФАЙЛАМИ (ПРОДОЛЖЕНИЕ)

```
  <Editor Grid.Row="1" x:Name="textEditor" />
    <ListView x:Name="filesList" Grid.Row="2" ItemSelected="FileSelect">
      <ListView.ItemTemplate>
        <DataTemplate>
          <TextCell Text="{Binding}">
            <TextCell.ContextActions>
              <MenuItem Text="Удалить" IsDestructive="True" Clicked="Delete" />
            </TextCell.ContextActions>
          </TextCell>
        </DataTemplate>
      </ListView.ItemTemplate>
    </ListView>
  </Grid>
</ContentPage>
```

ОБРАБОТЧИКИ СОБЫТИЙ

```
public partial class MainPage : ContentPage
    {
        string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);
        public MainPage()
        {
            InitializeComponent();
        }
```

ОБРАБОТЧИКИ СОБЫТИЙ

```
protected override void OnAppearing()
        {
            base.OnAppearing();
            UpdateFileList();
        }
```

ОБРАБОТЧИКИ СОБЫТИЙ

```
// сохранение текста в файл
        async void Save(object sender, EventArgs args)
        {
            string filename = fileNameEntry.Text;
            if (String.IsNullOrEmpty(filename)) return;
            // если файл существует
            if (File.Exists(Path.Combine(folderPath, filename)))
            {
                // запрашиваем разрешение на перезапись
                bool isRewrited = await DisplayAlert("Подтверждение", "Файл уже существует, перезаписать его?", "Да", "Нет");
                if (isRewrited == false) return;
            }
            // перезаписываем файл
            File.WriteAllText(Path.Combine(folderPath, filename), textEditor.Text);
            // обновляем список файлов
            UpdateFileList();
        }
```

ОБРАБОТЧИКИ СОБЫТИЙ

```
 void FileSelect(object sender, SelectedItemChangedEventArgs args)
        {
            if (args.SelectedItem == null) return;
            // получаем выделенный элемент
            string filename = (string)args.SelectedItem;
            // загружем текст в текстовое поле
            textEditor.Text = File.ReadAllText(Path.Combine(folderPath, (string)args.SelectedItem));
            // устанавливаем название файла
            fileNameEntry.Text = filename;
            // снимаем выделение
            filesList.SelectedItem = null;

        }
```

ОБРАБОТЧИКИ СОБЫТИЙ

```
void Delete(object sender, EventArgs args)
        {
            // получаем имя файла
            string filename = (string)((MenuItem)sender).BindingContext;
            // удаляем файл из списка
            File.Delete(Path.Combine(folderPath, filename));
            // обновляем список файлов
            UpdateFileList();
        }
```

ОБРАБОТЧИКИ СОБЫТИЙ

```
// обновление списка файлов
        void UpdateFileList()
        {
            // получаем все файлы
            filesList.ItemsSource = Directory.GetFiles(folderPath).Select(f => Path.GetFileName(f));
            // снимаем выделение
            filesList.SelectedItem = null;
        }
```

ПОЯСНЕНИЯ

- константой **Environment.SpecialFolder.Local ApplicationData**

- локальную папку приложения для хранения данных
- string folderPath - глобальную переменную, которая хранит путь к этой папке

ПОЯСНЕНИЯ

- При загрузке срабатывает метод UpdateFileList(), который загружает названия всех файлов из папки в список ListView:

ПОЯСНЕНИЯ

- С помощью
  метода Directory.GetFiles() возвращаем список всех файлов.
- Однако каждое название файла будет содержать полный путь с учетом всех внешних каталогов.
- И чтобы получить только непосредственно название файла, применяется метод Path.GetFileName, который из полного пути файла получет только его название.

ПОЯСНЕНИЯ

- Метод Save - обработчик нажатия кнопки будет проверять наличие файла.
- Если файл существует, то отображается диалоговое окно с требованием подтвердить действие.
- Если файл не существует, то он создается, и в него заносится текст из текстового поля Editor.

ПОЯСНЕНИЯ

- Стоит учитывать, что чтобы сохранить файл именно в той папке, с который мы работаем, необходимо использовать конкатенацию пути к папке и имени файла:
- **Path.Combine(folderPath, filename)**

ПОЯСНЕНИЯ

- сохранение производится с помощью метода File.WriteAllText:

- **File.WriteAllText(Path.Combine(folderPath, filename), textEditor.Text);**

ПОЯСНЕНИЯ

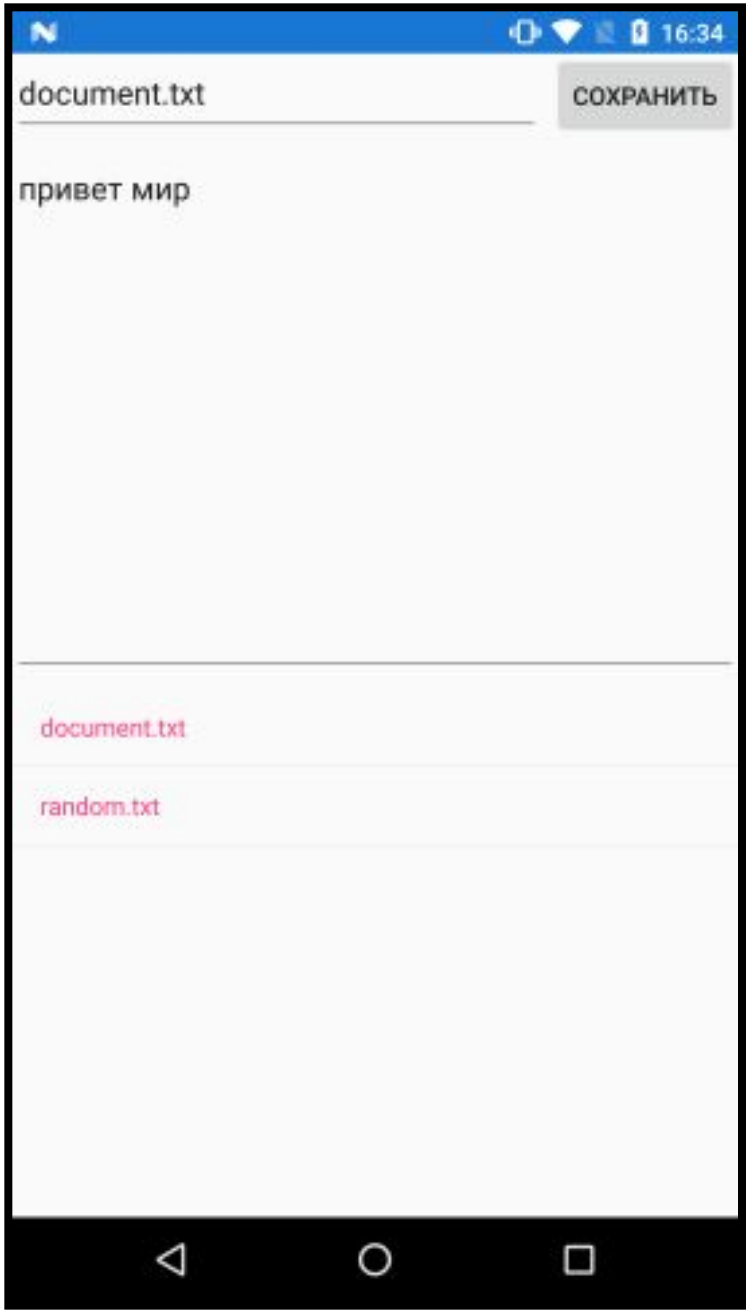
- Метод FileSelect() представляет обработчик выделения элемента в списке ListView. По выделению происходит загрузка текста в текстовое поле Editor.
- Для считывания текста файла применяется метод File.ReadAllText():
- **textEditor.Text = File.ReadAllText(Path.Combine(folderPath, (string)args.SelectedItem));**

ПОЯСНЕНИЯ

- метод Delete() является обработчиком нажатия на меню. То есть при сильном нажатии на элемент в списке ListView отображается данное контекстное меню из одного пункта, нажатие на которое приводит к удалению выделенного файла.

- **File.Delete(Path.Combine(folderPath, filename));**

ПОЯСНЕНИЯ


16:34
document.txt
СОХРАНИТЬ
привет мир
document.txt
random.txt

ЗАМЕЧАНИЯ

- 1. Возможны проблемы доступа к файлам, нужны права root
- 2. Библиотеки меняются, нужно быть внимательным, читать руководство
- Android.Environment
- System.Environment

ЗАДАНИЯ

- 1. Создать пример проекта согласно презентации, добавить комментарии в в код
- 2. Создать самостоятельно проект мобильного приложения для сохранения списка дел на неделю (аналог приложения Заметки):
- Просмотр заметок
- Добавление заметки
- Удаление выбранной заметки