Хранение данных в мобильных приложениях Xamarin

Типы хранилища

- Коллекция Properites в классе Application
- Настройки (Preferences) операционной системы
- Файлы
- Базы данных

Свойство Properites

- Свойство Properties класса Application представляет словарь, где каждому ключу с типом string сопоставляется некоторое значение

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"];
```

Удаление

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

Настройки приложения

В Xamarin Forms для работы с настройками мы можем использовать функциональность из пакета Xamarin Essentials, который добавляется в проект Xamarin Forms по умолчанию

Настройки приложения

Все настройки доступны через свойство Preferences, который предоставляет для работы с настройками следующие 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
- void Remove(key): удаляет настройку с ключом key
- void Clear(): удаляет все настройки
- bool ContainsKey(key): возвращает true, если имеется настройка с ключом key

Для сохранения применяется метод Set()

```
// 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");
```

Файлы

- Для работы с файлами предназначена пара классов File и FileInfo. С их помощью мы можем создавать, удалять, перемещать файлы, получать их свойства и многое другое

Полезные методы и свойства класса FileInfo

- CopyTo(path): копирует файл в новое место по указанному пути path
- Create(): создает файл
- Delete(): удаляет файл
- MoveTo(destFileName): перемещает файл в новое место
- Свойство Directory: получает родительский каталог в виде объекта DirectoryInfo
- Свойство DirectoryName: получает полный путь к родительскому каталогу
- Свойство Exists: указывает, существует ли файл
- Свойство Length: получает размер файла
- Свойство Extension: получает расширение файла
- Свойство Name: получает имя файла
- Свойство FullName: получает полное имя файла

Класс File реализует похожую функциональность с помощью статических методов

- **Copy()**: копирует файл в новое место
- **Create()**: создает файл
- **Delete()**: удаляет файл
- **Move**: перемещает файл в новое место
- **Exists(file)**: определяет, существует ли файл

Запись в файл и StreamWriter

- StreamWriter(string path): через параметр path передается путь к файлу, который будет связан с потоком
- StreamWriter(string path, bool append): параметр append указывает, надо ли добавлять в конец файла данные или же перезаписывать файл. Если равно true, то новые данные добавляются в конец файла. Если равно false, то файл перезаписываетсяя заново
- StreamWriter(string path, bool append, System.Text.Encoding encoding): параметр encoding указывает на кодировку, которая будет применяться при записи

Свою функциональность StreamWriter реализует через следующие методы

- int Close(): закрывает записываемый файл и освобождает все ресурсы
- void Flush(): записывает в файл оставшиеся в буфере данные и очищает буфер.
- Task FlushAsync(): асинхронная версия метода Flush
- void Write(string value): записывает в файл данные простейших типов, как int, double, char, string и т.д. Соответственно имеет ряд перегруженных версий для записи данных элементарных типов, например, Write(char value), Write(int value), Write(double value) и т.д.
- Task WriteAsync(string value): асинхронная версия метода Write
- void WriteLine(string value): также записывает данные, только после записи добавляет в файл символ окончания строки
- Task WriteLineAsync(string value): асинхронная версия метода WriteLine

Чтение из файла и StreamReader

Класс StreamReader позволяет нам легко считывать весь текст или отдельные строки из текстового файла.

Конструкторы класса StreamReader

- StreamReader(string path): через параметр path передается путь к считываемому файлу
- StreamReader(string path, System.Text.Encoding encoding): параметр encoding задает кодировку для чтения файла

Методы StreamReader

- void Close(): закрывает считываемый файл и освобождает все ресурсы
- int Peek(): возвращает следующий доступный символ, если символов больше нет, то возвращает -1
- int Read(): считывает и возвращает следующий символ в численном представлении. Имеет перегруженную версию: Read(char[] array, int index, int count), где array - массив, куда считываются символы, index - индекс в массиве array, начиная с которого записываются считываемые символы, и count - максимальное количество считываемых символов
- Task<int> ReadAsync(): асинхронная версия метода Read
- string ReadLine(): считывает одну строку в файле
- string ReadLineAsync(): асинхронная версия метода ReadLine
- string ReadToEnd(): считывает весь текст из файла
- string ReadToEndAsync(): асинхронная версия метода ReadToEnd