**28. Сохранение данных с помощью класса SharedPreferences**

**Теоретический материал**

Пять методов хранения данных Android:

1. SharedPreferences: хранить простые параметры конфигурации в виде карты;
2. ContentProvider: предоставлять частные данные приложения другим приложениям;
3. Файловое хранилище: хранится в виде потока ввода-вывода, который можно разделить на внутреннее и внешнее хранилище мобильного телефона (SD-карта и т. Д.), Который может хранить большие данные;
4. SQLite: легкая кроссплатформенная база данных, все данные хранятся в одном файле на телефоне, занимая небольшой объем памяти;
5. Сетевое хранилище: данные хранятся на сервере, и данные получаются при подключении к сети;

Sharedpreferences - это облегченный класс хранилища на платформе Android, используемый для хранения различной информации о конфигурации приложения. Его суть - XML-файл, который сохраняет данные в виде пар «ключ-значение». Файл хранится в / data / data // каталог shared\_prefs. С точки зрения глобальных переменных, его преимущество состоит в том, что он не создает проблем с приложениями, статическими переменными OOM (нехватка памяти) и нулевым указателем. Его недостатком является то, что эффективность не так высока, как у двух вышеуказанных методов.

**1. Получить SharedPreferences**

Чтобы использовать SharedPreferences для хранения данных, вам сначала нужно получить объект SharedPreferences. Android в основном предоставляет три метода получения объектов SharedPreferences.

**1.1 Метод getSharedPreferences () в классе Context:**

Этот метод получает два параметра. Первый параметр используется для указания имени файла SharedPreferences. Если указанный файл не существует, будет создан один. Второй параметр используется для указания режима работы. В основном существуют следующие режимы на выбор. MODE\_PRIVATE - это режим работы по умолчанию, и эффект такой же, как при прямой передаче 0.

Два режима MODE\_WORLD\_READABLE и MODE\_WORLD\_WRITEABLE объявлены устаревшими в Android 4.2.

**1.2 Метод getPreferences () в классе Activity:**

Этот метод похож на метод getSharedPreferences () в контексте, но он принимает только параметр режима работы, потому что при использовании этого метода имя текущего активного класса автоматически используется как имя файла SharedPreferences.

**1.3 Метод getDefaultSharedPreferences () в классе PreferenceManager:**

Это статический метод, который получает параметр Context и автоматически использует имя пакета текущего приложения в качестве префикса для имени файла Share

**Ход выполнения работы**

1. Компоненту textView дадим идентификатор tvResult, текст по умолчанию 0
2. Добавим кнопку и по событию Onclick пропишем функцию onClickAdd
3. Создаем функцию для увеличение числового значение в текстовом компоненте на 1 в class MainActivity и Объявим переменную

class MainActivity : AppCompatActivity() {  
 var counter = 0

fun onClickAdd(view: View)  
 {  
 counter++  
 tvResult.*text* = counter.toString()  
 saveData(counter)  
 }

1. Для сохранения значений текстового компонента, объявим переменную в class MainActivity

var pref : SharedPreferences? = null

1. И инициализируем ее в функции OnCreate

pref = getSharedPreferences("TABLE", Context.*MODE\_PRIVATE*)//определяем куда сохраним

1. И создаем функцию сохранения
2. fun saveData(res: Int)  
    {  
   val editor = pref?.edit()//определяем переменную в режим редактирования  
   editor?.putInt("counter", res)//добавление ключевого слова  
   editor?.apply()//применить изменения  
    }
3. необходимо чтобы при возобновлении работы приложения загружалось сохраненное значение, для этого в функции OnCreate пропишем , взять значение в переменную по ключевому слову

counter = pref?.getInt("counter",0)!!

counter = pref?.getString("test","0")!!

1. Для очистки данных создадим функцию fun deleteAll()  
    {  
    val editor = pref?.edit()  
    editor?.clear()  
    editor?.apply()  
    }
2. Для удаление данных по ключевому слову создадим функцию fun deleteIem(deletItem: String)  
    {  
    val editor = pref?.edit()  
    editor?.remove(deletItem)  
    editor?.apply()  
    }
3. Для работы этих двух функций пропишем функции

fun onClickClear(view: View)  
 {  
 deleteAll()  
 }  
 fun onClickDeleteItem(view: View)  
 {  
 deleteIem("test")  
 }

и это действие присвоим второй и третьей кнопке по событию OnClick

1. Чтобы посмотреть как работает удаление все-х данных и одного возьмём еще один textView дадим идентификатор tvRes2, текст по умолчанию 0

добавим переменную в class MainActivity

var test = "0"

И по событию OnCreate перепишем строку

counter = pref?.getString("test","0")!!

Как test = pref?.getString("test","0")!!

1. И добавим показ значений

tvResult.*text* = counter.toString()  
tvRes2.*text* = test

1. И в функцию fun saveData(res: Int)  
   добавим перед строкой применить изменения

editor?.putString("test", "Test")//добавить текст