Android1.Lesson07

Урок 07. Хранение данных в Android

Домашнее задание

Вопросы?

Структура урока

- Сохранение промежуточного состояния Activity
- Работа с настройками приложения
- Внутренняя и внешняя память устройства
  - Доступ к внутренней памяти устройства
  - Получение состояния карты памяти
  - Работа с внешней памятью устройства

Состояние Activity

Сохранение промежуточного состояния Activity, работа с объектом Bundle

Жизненный цикл Activity

- Activity может быть остановлено и, затем, уничтожено в любой момент
- При изменении ориентации экрана, Activity так же создается заново
- Часто возникает необходимость сохранить промежуточное состояние Activity для предотвращения потери данных пользователя

Состояние Activity

- Состояние Activity – состояние элементов управления и некоторые значения, определенные в классе Activity приложения
- Android позволяет сохранять состояние Activity перед его деактивацией (*onSaveInstanceState*) и восстановить при последующей активации (*onCreate*, *onRestoreInstanceState*)

Состояние Activity

- onSaveInstanceState – вызывается, когда Activity уничтожается принудительно, например из за нехватки ресурсов или при изменении конфигурации устройства (смена ориентации экрана)
- onRestoreInstanceState – вызывается после метода *onStart* при повторной инициализации Activity (если при уничтожении вызывался метод *onSaveInstanceState*), но в большинстве случаев можно выполнять восстановление данных в методе *onCreate*

Объект Bundle

- Для работы с состоянием Activity используется объект типа Bundle, в котором можно хранить пары «Ключ-Значение»
- В метод *onSaveInstanceState* объект этого класса передается в качестве параметра для сохранения состояния Activity (в виде пар «Ключ-Значение»)
- В методы *onCreate* и *onRestoreInstanceState* объект этого класса передается в качестве параметра для восстановления состояния Activity

Объект Bundle

- По умолчанию, система использует объект типа Bundle для сохранения и восстановления информации о всех объектах View в Activity (например содержимое EditText)
- Объект Bundle используется только в том случае, если уничтожение Activity было инициировано системой, а не пользователем (то есть, если пользователь закрыл приложение, объект Bundle не будет хранить данные)
- Если уничтожение Activity было инициировано пользователем (например кнопкой «Назад»), объект Bundle в методе *onCreate* будет равен *null*

Пример кода

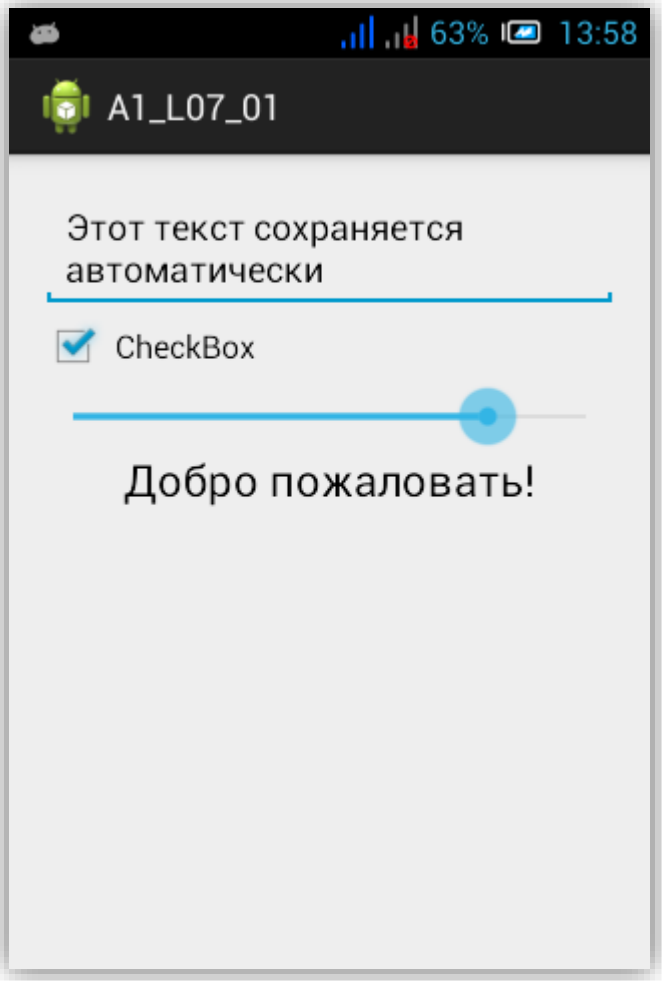
Сохранение данных

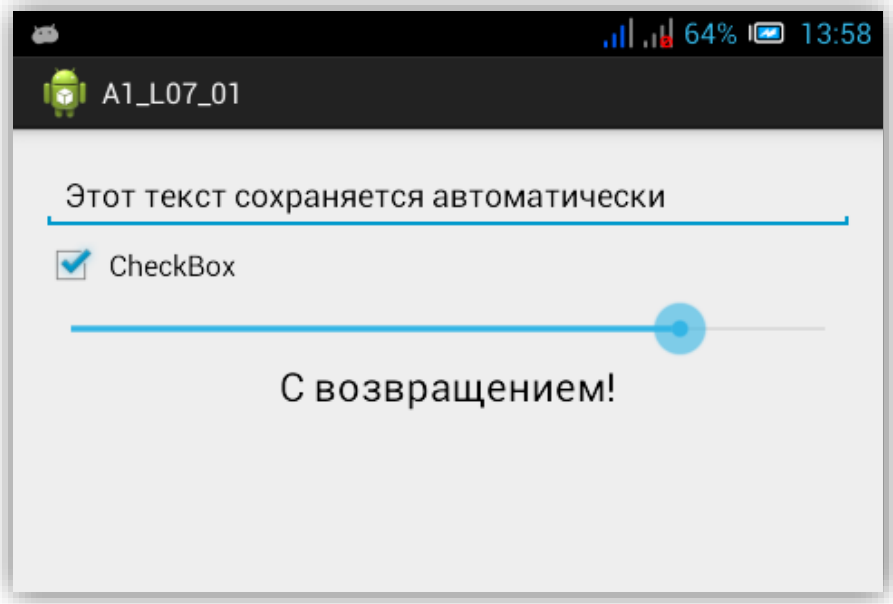
```
/**
 * Вызывается, перед принудительным уничтожением Активити и позволяет
 * сохранить его промежуточное состояние (любые данные).
 *
 * @param outState объект, в который можно поместить любые данные в
 * формате имя-значение (key-value) для сохранения.
 *
 * */
@Override
protected void onSaveInstanceState(Bundle outState) {

    /* Значение типа int */
    outState.putInt(STATE_ANY_INT, ++anyStateInt);

    /* Значение типа boolean */
    outState.putBoolean(STATE_WAS_RESTORED, true);

    /*
     * Вызов метода предка (класс Activity) для сохранения состояния всех
     * UI элементов
     * */
    super.onSaveInstanceState(outState);

}
```

Восстановление данных

```
/*
 * Восстановление промежуточного состояния Активити (если оно было
 * сохранено ранее). Восстановление состояния Активити можно так же
 * выполнить в методе onRestoreInstanceState(), который будет вызван
 * после метода onStart() в том случае, если состояние было сохранено
 * ранее.
 * */
if (savedInstanceState != null) {

    /* Значение типа int */
    anyStateInt = savedInstanceState.getInt(STATE_ANY_INT);

    /* Значение типа boolean */
    wasRestored = savedInstanceState.getBoolean(STATE_WAS_RESTORED,
            true);
}

/* Формирование содержимого tvText */
if (!wasRestored) {
    tvText.setText("Добро пожаловать!");
} else
    tvText.setText("С возвращением!");
```

Пример приложения





Состояние Activity

Вопросы?

Настройки приложения

Работа с настройками приложения

Shared Preferences

- Класс *SharedPreferences* предоставляет общий фреймворк позволяющий сохранять и извлекать (загружать) пары «Ключ-Значение» для примитивных типов данных
- Данные объекта *SharedPreferences* хранятся в файлах в защищенной области памяти (папка приложения)
- Любые данные, сохраненные через объект *SharedPreferences*, будут доступны не зависимо от состояния приложения (было ли оно перезапущено или уничтожено и тд) и будут автоматически удалены при удалении приложения из системы

Shared Preferences

- Работать с объектом *SharedPreferences* можно из любой точки приложения, без привязки к методам жизненного цикла
- Класс *SharedPreferences* используется для хранения настроек приложения
- Класс *SharedPreferences* позволяет использовать разные хранилища данных (разные файлы)

Shared Preferences

- Получить объект *SharedPreferences* можно вызвав метод:
  - *getSharedPreferences()* – используется, когда нужно указать имя файла данных (например если таких файлов несколько). Имеет два параметра, имя файла данных и метод доступа
  - *getPreferences()* – используется, когда нужно выбрать файл данных по умолчанию (для текущего Activity). В качестве имени файла используется имя класса Activity

Пример кода

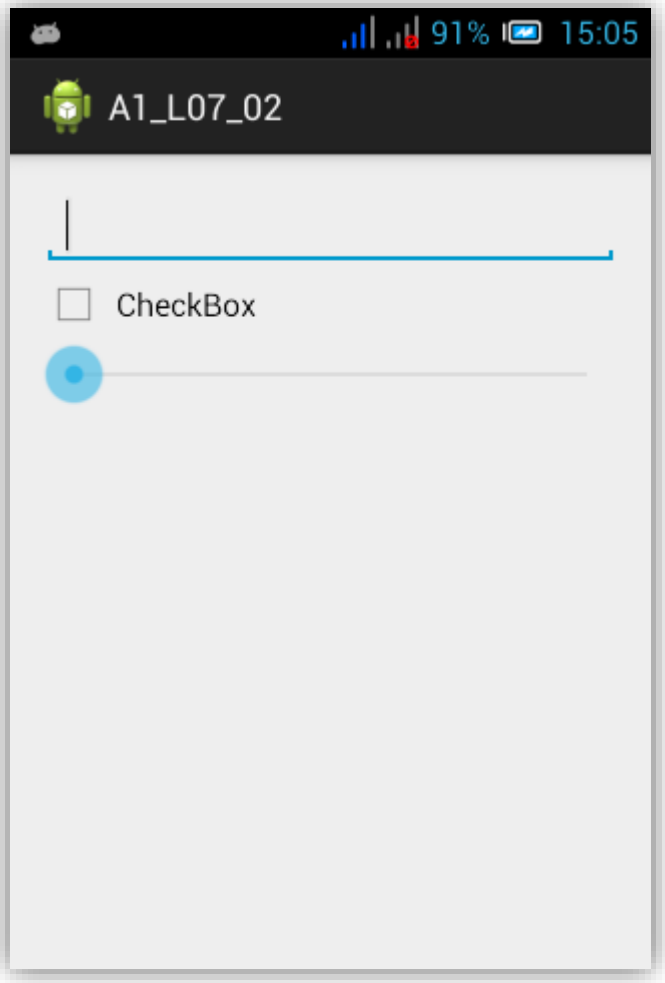
Сохранение данных

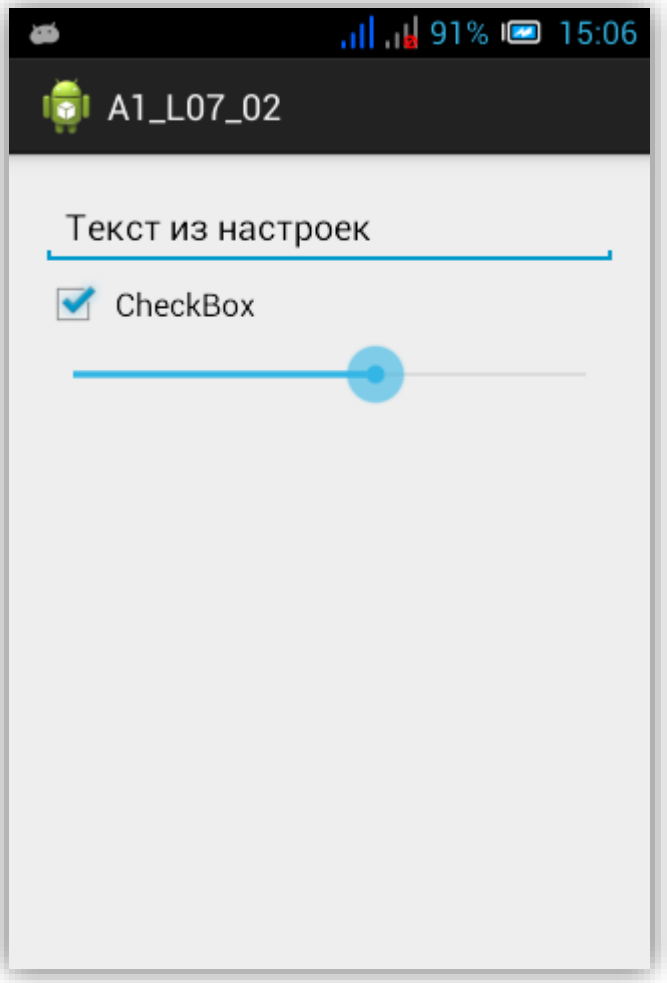
```
/* Ссылка на объект настроек по умолчанию */
SharedPreferences options = this.getPreferences(MODE_PRIVATE);

/* Ссылка на объект редактора настроек */
SharedPreferences.Editor editor = options.edit();

/*
 * Сохранение данных компонентов UI
 * */
editor.putBoolean(PREF_CHECH_BOX, cbCheck.isChecked());
editor.putInt(PREF_SEEK_BAR, sbSeek.getProgress());

/* Применить изменения */
editor.commit();

/* Ссылка на объект настроек с именем MyPrefFile */
options = this.getSharedPreferences(PREF_NAME, MODE_PRIVATE);

/* Ссылка на объект редактора настроек */
editor = options.edit();

/*
 * Сохранение данных компонентов UI
 * */
editor.putString(PREF_EDIT_TEXT, edText.getText().toString());

/* Применить изменения */
editor.commit();

/* Вызов метода предка (класс Activity) */
super.onStop();
```

Восстановление данных

```
/* Ссылка на объект настроек по умолчанию */
SharedPreferences options = this.getPreferences(MODE_PRIVATE);

/*
 * Загрузка данных для компонентов UI
 * */
int progress = options.getInt(PREF_SEEK_BAR, 0);
sbSeek.setProgress(progress);

cbCheck.setChecked(options.getBoolean(PREF_CHECH_BOX, false));

/* Ссылка на объект настроек с именем MyPrefFile */
options = this.getSharedPreferences(PREF_NAME, MODE_PRIVATE);

/*
 * Загрузка данных для компонентов UI
 * */
String text = options.getString(PREF_EDIT_TEXT, "");
edText.setText(text);
```

Пример приложения





Вопросы?

Работа с файлами

Работа с файлами, внутренняя и внешняя память

Внутренняя память

- Можно сохранять файлы во внутренней памяти устройства
- По умолчанию эти файлы не доступны другим приложениям в системе и пользователю
- При удалении приложения эти файлы будут удалены автоматически
- Общий алгоритм работы:
  - Открыть файл на чтение или запись
  - Прочитать/записать данные
  - Закрыть файл

Внутренняя память

Запись данных

```
/* Текст для сохранения в файл */
String text = edText.getText().toString();

/* Переменная для работы с файлом */
FileOutputStream fos = null;

/* Работа с файлом */
try {

    /* Открывает файл в режиме записи (FileNotFoundException) */
    fos = this.openFileOutput(FILE_NAME, Context.MODE_PRIVATE);

    /* Запись в файл (IOException) */
    fos.write(text.getBytes());

    /* Закрываем файл (IOException) */
    fos.close();

} catch (Exception e) {
    /* Запись в лог */
    e.printStackTrace();
}
```

Чтение данных

```
/* Переменная для работы с файлом */
FileInputStream fis = null;

/* Работа с файлом */
try {

    /* Открывает файл в режиме чтения (FileNotFoundException) */
    fis = this.openFileInput(FILE_NAME);

    /* Определение размера данных в байтах (IOException) */
    int bufSize = fis.available();

    /* Подготовка буфера для данных из файла */
    byte[] buffer = new byte[bufSize];

    /* Чтение из файла (IOException) */
    fis.read(buffer);

    /* Создание строки на основе данных из файла */
    String text = new String(buffer);
    edText.setText(text);

    /* Закрываем файл (IOException) */
    fis.close();

} catch (Exception e) {
    /* Запись в лог */
    e.printStackTrace();
}
```

Внешняя память

- Android устройства поддерживают работу с внешним накопителем (отдельная или встроенная карта памяти), который можно использовать для хранения файлов
- Такие файлы обычно доступны другим приложениям и пользователю, а так же могут быть видны с компьютера, при подключении устройства
- Внешний накопитель может оказаться не доступным для использования в приложении (пользователь извлек карту памяти, устройство находиться в режиме USB накопителя)

Внешняя память

- При удалении приложения, файлы, созданные им для общего доступа и находящиеся на внешней памяти автоматически не удаляются
- Для работы с внешней памятью приложение должно иметь разрешение:
  - *android.permission.WRITE_EXTERNAL_STORAGE*
- Перед операциями чтения/записи необходимо проверить состояние внешнего накопителя (его доступность)

Состояния внешнего накопителя

- MEDIA_BAD_REMOVAL
- MEDIA_CHECKING
- MEDIA_MOUNTED
- MEDIA_MOUNTED_READ_ONLY
- MEDIA_NOFS
- MEDIA_REMOVED
- MEDIA_SHARED
- MEDIA_UNMOUNTABLE
- MEDIA_UNMOUNTED

Получение состояния накопителя

Запись и чтение

```
/**
 * Проверка доступности внешней памяти для чтения и записи
 * */
public boolean isExternalStorageWritable() {

    /* Получение текущего состояния внешней памяти */
    String state = Environment.getExternalStorageState();

    /* Проверка текущего состояния и возврат значения */
    return Environment.MEDIA_MOUNTED.equals(state);

}
```

Чтение

```
/**
 * Проверка доступности внешней памяти для чтения
 * */
public boolean isExternalStorageReadable() {

    /* Получение текущего состояния внешней памяти */
    String state = Environment.getExternalStorageState();

    /* Проверка текущего состояния и возврат значения */
    return Environment.MEDIA_MOUNTED.equals(state) ||
            Environment.MEDIA_MOUNTED_READ_ONLY.equals(state);
}
```

Внешняя память

Запись данных

```
/* Текст для сохранения в файл */
String text = edText.getText().toString();

/* Переменная для работы с файлом */
FileWriter fw = null;

/* Работа с файлом */
try {

    /* Формирование пути к файлу */
    String fullName = Environment.getExternalStorageDirectory().
            getAbsolutePath() + "/" + FILE_NAME;

    /* Открывает файл в режиме записи (IOException) */
    fw = new FileWriter(fullName);

    /* Запись в файл (IOException) */
    fw.write(text);

    /* Закрываем файл (IOException) */
    fw.close();

    edText.setText(fullName);

} catch (Exception e) {
    /* Запись в лог */
    e.printStackTrace();
}
```

Чтение данных

```
/* Переменная для работы с файлом */
FileReader fr = null;

/* Работа с файлом */
try {

    /* Формирование пути к файлу */
    String fullName = Environment.getExternalStorageDirectory().
            getAbsolutePath() + "/" + FILE_NAME;

    /* Открывает файл в режиме чтения (FileNotFoundException) */
    fr = new FileReader(fullName);

    /* Подготовка буфера для данных из файла */
    CharBuffer buffer = CharBuffer.allocate(64);

    /* Чтение из файла (IOException) */
    fr.read(buffer);

    /* Создание строки на основе данных из файла */
    String text = new String(buffer.array());
    edText.setText(text);

    /* Закрываем файл (IOException) */
    fr.close();

} catch (Exception e) {
    /* Запись в лог */
    e.printStackTrace();
}
```

Вопросы?

Домашнее задание

- Разобраться с материалами урока
- Написать приложение «Заметки»:
  - Просмотр списка заметок (ListView)
  - Удаление заметки из списка
  - Изменение текста заметки
  - Добавление новой заметки