08

Сеть
Игра в волейбол

Nikolay Moskvin
moskvin@sibext.com

12 ноября 2013 г.

Содержание

1 Игра в волейбол

2 Работа с камерой

3 Новые возможности у камеры

Игра в волейбол

sibext company

- URL: `http://volley.sibext.com/`
- Action: POST
- Request:

```
1 {
2 direction = 0|1
3 }
```

- Response:

```
1 { "direction"= 1|0,
2 "place" = "center"|"right"|"left"|"near
     grid",
3 "result" = "goal"|"caught"|"out" }
```

Снаряжение для волейбола

sibext company

- **LruCache** – организация кэша, на основе least recently used
- **RequestQueue** – используется для организации thread pool
- **JsonObjectRequest** – позволяет создать http запрос на получения JSON
  - **method** – классический HTTP метод (GET, POST, PUT...)
  - **jsonRequest** – входные данные в виде JSON
  - **listener** – callback, который придет при получении ответа от сервера
  - **errorListener** – callback, который если операция не удалась
- **JSONObject** – основной объект JSON
- Github: `https://github.com/sibext/VolleyExample`

Волейбол через Gson

sibext company

- Добавляем в pom.xml

```
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.2.2</version>
</dependency>
```

- Расширяем класс **Request**
  - **deliverResponse** – осуществляем доставку уже разобранного результата в наш собственный callback
  - **parseNetworkResponse** – реализуем разбор с помощью NetworkResponse, необходимо воспользоваться методами **Response.success** и **gson.fromJson**
  - **getBody** – реализуем передачу входных данных, необходимо воспользоваться методами **getBytes** и **gson.toJson**
- **VolleyRequest и VolleyResponse** – необходимо определить модельки для входных и выходных данных
- Потребуется модификация класса listener

Камера

Необходимые условия для использования камеры в компонентах

- **Обязательные требования**
  - Ваш компонент должен запросить разрешения

```
<uses-permission android:name="android.permission.CAMERA" />
```

  - Ваш компонент должен заявить о использовании функций камеры

```
<uses-feature android:name="android.hardware.camera" />
```

  - Для сохранения итоговых изображений или видео на SD card

```
<uses-permission
  android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Целесообразность реализовывать камеру:

- Если нужно сделать быстрый снимок или снять видеоклип, то можно:
  - **Составить Intent для камеры**. Может быть двух видов:
    - **MediaStore.ACTION_IMAGE_CAPTURE** – запрашивает снимок у уже существующего приложения
    - **MediaStore.ACTION_VIDEO_CAPTURE** – запрашивает видео у уже существующего приложения
  - **Запустить созданный Intent**. Необходимо использовать метод startActivityForResult()
  - **Получить готовый снимок или видео**. Необходимо переопределить метод onActivityResult()

Получаем фото-снимок

sibext company

Минимум усилий для получения видео, указываем в exta полный путь, куда сохранить фото:

- **MediaStore.EXTRA_OUTPUT** – обязательно должен иметь тип Uri. Параметр необязательный
- **Пример:**

```
private static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 100;
private Uri fileUri;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    File imagesFolder =
      new File(Environment.getExternalStorageDirectory(), "images");
    imagesFolder.mkdirs();
    File image = new File(imagesFolder, "photo.jpg");
    fileUri = Uri.fromFile(image);
    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);

    startActivityForResult(intent, CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
}
```

Получаем видео-ролик

sibext company

Минимум усилий для получения роликов необходимо настроить:

- **MediaStore.EXTRA_OUTPUT** – полный путь до файла с роликов, обязательно должен иметь тип Uri. Параметр необязательный
- **MediaStore.EXTRA_VIDEO_QUALITY** – качество видео. 1 – наивысшее, 0 – наихудшее.
- **MediaStore.EXTRA_DURATION_LIMIT** – задается длительность ролика в секундах.
- **MediaStore.EXTRA_SIZE_LIMIT** – задается максимальный размер видео-файла в байтах.
- **Пример:**

Получаем видео-ролик

sibext company

- **Пример:**

```
private static final int CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE = 200;
private Uri fileUri;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    Intent intent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
    File imagesFolder =
      new File(Environment.getExternalStorageDirectory(), "video");

    imagesFolder.mkdirs();

    File image = new File(imagesFolder, "clip.mp4");
    fileUri = Uri.fromFile(image);
    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
    intent.putExtra(MediaStore.EXTRA_VIDEO_QUALITY, 1);

    startActivityForResult(intent, CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE);
```

Получения результатов с камеры

- Необходимо реализовать метод **onActivityResult()**.
  Пример:

```
@Override
protected void onActivityResult(int requestCode,
    int resultCode, Intent data) {
    if (requestCode == CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            Toast.makeText(this, "Image saved to:\n" +
                    data.getData(), Toast.LENGTH_LONG).show();
        } else if (resultCode == RESULT_CANCELED) {
            // User cancelled the image capture
        }
    }
    if (requestCode == CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            Toast.makeText(this, "Video saved to:\n" +
                    data.getData(), Toast.LENGTH_LONG).show();
        } else if (resultCode == RESULT_CANCELED) {
            // User cancelled the video capture
        }
    }
}
```

- Как только прозвучит этот метод с положительным результатом, ваше приложение может использовать видео или фото в вашем приложении. (если не задан путь, то будет сохранено **data.getData()**)

Возводим собственное поведение для камеры

sibext company

Основные шаги для этого:

- **Определение и доступ к камере** – кусочек кода, который проверяет есть ли камера на устройстве и возвращает экземпляр камеры, если есть доступ к камере.
- **Создание класса для предпросмотра** – используются класс **SurfaceView** и интерфейс **SurfaceHolder** для создания “живой” картинки с камеры
- **Верстка layout** – классическая верстка в xml
- **Настройка слушателей** – необходимо подцепить необходимые кнопки на прослушку событий с камеры
- **Сохранения результатов** – кусочек кода, который делает захват изображения или видео с камеры и сохраняет их
- **Освобождение камеры** – после использования не забываем освободить камеру надлежащим способом

Внимание

Не забывайте освобождать камеру с помощью метода **Camera.release** после использования. Если Ваше приложение

Возводим собственное поведение для камеры

sibext company

Внимание

Не забывайте освобождать камеру с помощью метода **Camera.release** после использования. Если Ваше приложение не отпустит камеру после окончания работы компонента, то на последующие попытки получить доступ к камере вы получите отказ (устройство уже занято)! Все другие приложения на устройстве тоже не смогут использовать больше камеру, пока Вы не перезагрузите устройство

Проверяем доступ к камере

sibext company

- Проверяем есть ли возможность работать с камерой:

```
private boolean checkCameraHardware(Context context) {
    if (context.getPackageManager()
        .hasSystemFeature(PackageManager.FEATURE_CAMERA)){
        return true;
    } else {
        return false;
    }
}
```

- Можем проверить на количество камер на одном устройстве **Camera.getNumberOfCameras()**

Доступ к камере

sibext company

- Необходимо обернуть открытие камеры в try - catch секцию. Открываем камеру по умолчанию:

```
public static Camera getCameraInstance(){
    Camera c = null;
    try {
        c = Camera.open();
    }catch (Exception e){
        // Camera is not available (in use or does not exist)
    }
    return c;
}
```

- Если Вам требуется какая-то конкретная камера, то можете воспользоваться методом с индексом **Camera.open(int)**

Проверка функциональности камеры

sibext company

- Проверяем камеру на пригодность с помощью методов:
  - **Camera.getParameters()** – вернет все доступные возможности для камеры

```
boolean supportedMono = false;
camera.getParameters().setFocusMode(Camera.Parameters.FOCUS_MODE_AUTO);
try { for (String effect: camera.getParameters()
      .getSupportedColorEffects()) {
    if (effect.equals(
      android.hardware.Camera.Parameters.EFFECT_MONO)) {
      supportedMono = true;
      break;
    }
  } } catch (Exception e) {
     Log.e(TAG, "Unsupported mono features on this camera", e);
}
```

  - **Camera.getCameraInfo()** – определяем тыльная или центральная камера на устройстве

```
Camera.CameraInfo cameraInfo = new Camera.CameraInfo();
for (int idx = 0; idx < Camera.getNumberOfCameras(); idx++) {
  Camera.getCameraInfo(camIdx, cameraInfo);
  if (cameraInfo.facing == Camera.CameraInfo.CAMERA_FACING_BACK) {
      try { camera = Camera.open(idx);
            Log.i(TAG, "Camera found and open: " + idx);
            break;
      } catch (RuntimeException e) {
        Log.e(TAG, "Camera failed to open ", e);
      }
  }}
```

Создание предпросмотра

sibext company

- Создаем SurfaceView Основные методы:
  - **surfaceCreated** – вызывается один раз при создании, мы указываем камере где у нас будет показан предпросмотр.
  - **surfaceDestroyed** – вызывается, когда объект уже не нужен, здесь необходимо освободить камеру
  - **surfaceChanged** – в данном методе необходимо перезапускать предпросмотр
  - Пример: `http://developer.android.com/guide/topics/media/camera.html#camera-preview`
- Верстаем layout
- Запускаем предпросмотр в Activity
  - создаем камеру:

```
camera = getCameraInstance();
```

  - создаем предпросмотр и устанавливаем контент нашей Activity:

```
2  preview = new CameraPreview(this, mCamera);
   FrameLayout preview = (FrameLayout) findViewById(R.id.preview);
   preview.addView(preview);
```

Захват изображения

sibext company

- Реализуем слушателя **PictureCallback**
- Используем **Camera.takePicture()** в том месте где пользователь хочет получить фото

```
private PictureCallback pictureListener = new PictureCallback() {
 @Override
 public void onPictureTaken(byte[] data, Camera camera) {
     File pictureFile = getOutputMediaFile(MEDIA_TYPE_IMAGE);
     if (pictureFile == null){
         Log.d(TAG, "Error creating media file, check permissions: " +
             e.getMessage());
         return;
     }

     try {
         FileOutputStream fos = new FileOutputStream(pictureFile);
         fos.write(data);
         fos.close();
     } catch (FileNotFoundException e) {
         Log.d(TAG, "File not found: " + e.getMessage());
     } catch (IOException e) {
         Log.d(TAG, "Error accessing file: " + e.getMessage());
     }
 }
};
```

Захват изображения

sibext company

- Пример вызова захвата фото

```
Button captureButton = (Button) findViewById(id.button_capture);
captureButton.setOnClickListener(
  new View.OnClickListener() {
      @Override
      public void onClick(View v) {
          camera.takePicture(
            null, // for image capture moment
            null, // for raw (uncompressed) image data
            pictureListener);
      }
  }
);
```

Определение лиц у людей

Необходимы шаги, для того, чтобы использовать эту возможность:

- Проверяем устройство поддерживает возможность определения лиц, появилась с Android 4.0 (API Level 14)
- Создаем слушателя для считывания лиц на экране

```
class FriendsFaceDetectionListener
      implements Camera.FaceDetectionListener {

    @Override
    public void onFaceDetection(Face[] faces, Camera camera) {
        if (faces.length > 0){
            Log.d(TAG, "face detected: "+ faces.length +
                    " Face 1 Location X: " + faces[0].rect.centerX() +
                    "Y: " + faces[0].rect.centerY() );
        }
    }
}
```

- Прицепляем слушателя к камере

```
camera.setFaceDetectionListener(new FriendsFaceDetectionListener());
```

- Запускаем наше определение лиц после каждого перезапуска предпросмотра камеры

Запуск определения лиц

```
public void startFaceDetection(){
    Camera.Parameters params = camera.getParameters();

    // start face detection only *after* preview has started
    if (params.getMaxNumDetectedFaces() > 0){
        // camera supports face detection, so can start it:
        camera.startFaceDetection();
    }
}
```

Внимание

Не запускайте **startFaceDetection()** метод в **onCreate()**.
Вызываем его только после КАЖДОГО вызова **startPreview()**
Обычно Вам придется вызвать его в методах **surfaceChanged** и **surfaceCreated**

Запуск определения лиц

```
public void startFaceDetection(){
    Camera.Parameters params = camera.getParameters();

    // start face detection only *after* preview has started
    if (params.getMaxNumDetectedFaces() > 0){
        // camera supports face detection, so can start it:
        camera.startFaceDetection();
    }
}
```

Внимание

Не запускайте **startFaceDetection()** метод в **onCreate()**. Вызываем его только после КАЖДОГО вызова **startPreview()** Обычно Вам придется вызвать его в методах **surfaceChanged** и **surfaceCreated**