桂林电子科技大学

序号：

课号：1922137

**Android应用开发**  实验报告

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 实验十三 音乐播放器四 | | | | | | | |  | 辅导员意见：  成绩 辅导员  签 名 |
| 院 系 | 计算机与信息安全学院 | | | 专业 | | 计算机科学与技术 | | |
| 学 号 | 1800301037 | | | 姓名 | | 禤成伟 | | |
| 实验日期 | 2020 | 年 | 6 | | 月 | | 1 | 日 |
|  |  | | | | | | | |

|  |
| --- |
| **一、实验目的**  1. 掌握绑定服务 (Bound service) 的定义⽅式及⽤法；  2. 掌握⾃定义⼴播及其⽤法；  3. 掌握线程间通信的⽅法； |
| **二、实验内容**  1. 为其他组件提供访问 MusicService 实例提供方法 ：在 MusicService 类中定义名为 MusicService?Binder 的内部类，该类继承? Binder 类。此在 MusicServiceBinder 类中，定义了名为 getService 的方法返回当前 MusicService 类实例。  2. 获得向其他组件提供绑定服务对象的 IBinder 实例：重写服务中的 onBind() 方法，获取在 MusicService 中定义的私有成员 mBinder。  3. MusicService 暴露 pause()、play()、getDuration()、getCurrentPosition()、isPlaying() 接口给其他组件使用。  4. 绑定及解绑 MusicService 服务。  5.使用多线程实时更新音乐播放进度状态。  6. 自定义 MusicReceiver 广播MusicService 发送 AC\_TION\_MUSIC\_START ⼴播，MainActivity 接收该⼴播，启动 MusicProgress Runnable ⼦线程获取⾳乐播放进度，Handler 处理⼦线程消息更新 ProgressBar进度条显⽰播放进度。 |
| 图6.1音乐播放器  **三、实验总结**  问题：运行时一片空白。  解决方法:其实没出什么问题，一片空白是因为我用虚拟机运行，虚拟机里面没有音乐，所以一片空白。之后改用手机运行，出现了实验指导书上的效果。 |
| **四、实验代码**   |  |  | | --- | --- | | 代码6.1: MusicServiceBinder   |  | | --- | | private final IBinder mBinder = new MusicServiceBinder();  public class MusicServiceBinder extends Binder {  MusicService getService() {  return MusicService.this;  } } | |   代码 6.2: 重写 onBind ⽅法   |  | | --- | | @Override public IBinder onBind(Intent intent) {  return mBinder; } |   代码 6.3: 实现 MusicService 对外⽅法   |  | | --- | | /\*\* method for clients \*/ public void pause() {  if (mMediaPlayer != null && mMediaPlayer.isPlaying()) {  mMediaPlayer.pause();  } }  public void play() {  if (mMediaPlayer != null) {  mMediaPlayer.start();  } }  public int getDuration() {  int duration = 0;   if (mMediaPlayer != null) {  duration = mMediaPlayer.getDuration();  }   return duration; } public int getCurrentPosition() {  int position = 0;   if (mMediaPlayer != null) {  position = mMediaPlayer.getCurrentPosition();  }   return position; }  public boolean isPlaying() {   if (mMediaPlayer != null) {  return mMediaPlayer.isPlaying();  }  return false; } |   代码 6.4: 绑定及解绑 MusicService 服务   |  | | --- | | @Override protected void onStart() {  super.onStart();   Intent intent = new Intent(MainActivity.this ,  MusicService.class);  bindService(intent , mConn , Context.BIND\_AUTO\_CREATE); }  @Override protected void onStop() {  unbindService(mConn);  mBound = false;   super.onStop(); } |   代码6.5: 调⽤ MusicService 的 pause() 及 play() ⽅法   |  | | --- | | //设置监听事件 if (ivPlay != null) {  ivPlay.setOnClickListener(this); }  @Override public void onClick(View view) {  //控制播放和暂停图标切换 以及在暂停时关闭后台播放服务  if (view.getId() == R.id.iv\_play) {  mPlayStatus = !mPlayStatus;   if (mPlayStatus == true) {  mService.play();  ivPlay.setImageResource(  R.drawable.ic\_pause\_circle\_outline\_black\_24dp);  } else {  mService.pause();  ivPlay.setImageResource(  R.drawable.ic\_play\_circle\_outline\_black\_24dp);  }  } } |   代码6.6: 更新音乐播放进度   |  | | --- | | private Handler mHandler = new Handler(Looper.getMainLooper()) {   public void handleMessage(Message msg) {  switch (msg.what) {  case UPDATE\_PROGRESS:  int position = msg.arg1;  pbProgress.setProgress(position);  break;  default:  break;  }  } };    class MusicProgressRunnable implements Runnable {  public MusicProgressRunnable() {  }   @Override  public void run() {  boolean mThreadWorking = true;  while (mThreadWorking) {  try {  if (mService != null) {  int position =  mService.getCurrentPosition();  Message message = new Message();  message.what = UPDATE\_PROGRESS;  message.arg1 = position;  mHandler.sendMessage(message);  }  mThreadWorking = mService.isPlaying();  Thread.sleep(100);  } catch (InterruptedException ie) {  ie.printStackTrace();  }  }  } } |   代码6.7: 注册 MusicReceiver ⼴播   |  | | --- | | @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  musicReceiver = new MusicReceiver();  IntentFilter intentFilter = new IntentFilter();  intentFilter.addAction(ACTION\_MUSIC\_START);  intentFilter.addAction(ACTION\_MUSIC\_STOP);  registerReceiver(musicReceiver, intentFilter);  }  @Override  protected void onDestroy() {  unregisterReceiver(musicReceiver);  super.onDestroy();  } | |  |   代码6.8: 发送 ACTION\_MUSIC\_START ⼴播   |  | | --- | | @Override public int onStartCommand(Intent intent,  int flags, int startId) {  String data = intent.getStringExtra(  MainActivity.DATA\_URI);  Uri dataUri = Uri.parse(data);   if (mMediaPlayer != null) {  try {  mMediaPlayer.reset();  mMediaPlayer.setDataSource(  getApplicationContext(),  dataUri);  mMediaPlayer.prepare();  mMediaPlayer.start();    Intent musicStartIntent =  new Intent(MainActivity.ACTION\_MUSIC\_START);  sendBroadcast(musicStartIntent);    } catch (IOException ex) {  ex.printStackTrace();  }  }   return super.onStartCommand(intent, flags, startId); } | |