数据的获取存储及上传

数据的获取

程序中用到的属性

具体属性字段如下所示：

```
private static TextView tvStatus;                    // 显示手机与传感器之间的状态
private static String mHxMName = null;               // 记录传感器的名称
private static String mHxMAddress = null;            // 记录传感器的地址
// 移动设备的本地的蓝牙适配器，通过该蓝牙适配器可以对蓝牙进行基本操作
private static BluetoothAdapter mBluetoothAdapter = null;

// 设置和管理蓝牙和传感器设备之间的连接
private static HxmService mHxmService = null;

// 从传感器中读取相应的数据
private static HxmRead hxmRead;

// 将这个类设计为单例模式
private static AtyHeartRate instance;
```

状态的初始化

这个方法总体上的功能是初始化程序的各个配置，包括单例模式的初始化，传感器服务是否已经启动，例如判断手机的蓝牙服务是否启动和尝试着连接传感器设备。

具体代码实现如下：

```
instance = this;
if(mHxmService == null) {
    tvStatus = (TextView) findViewById(R.id.status);
    tvStatus.setText(R.string.initializing);
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

    // If the adapter is null, then Bluetooth is not supported
    if(mBluetoothAdapter == null) {
        //Bluetooth needs to be available on this device, and also enabled.
        tvStatus.setText(R.string.noBluetooth);
```

```
            } else {
                if(!mBluetoothAdapter.isEnabled()) {
                    tvStatus.setText(R.string.btNotEnabled);
                } else {
                    tvStatus.setText(R.string.connecting);

                    connectToHxm();
                }
            }
        } else {
            instance.onResume();
        }
```

连接到传感器

connectToHxm()函数将设置开始连接和管理传感器数据流之间的逻辑，具体代码如下：

```
  private void connectToHxm() {

   mStatus.setText(R.string.connecting);
    mState=getResources().getString(R.string.connecting);

  if (mHxmService == null)
   setupHrm();

  if ( getFirstConnectedHxm() ) {
    BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(mHxMAddress);
    mHxmService.connect(device);   // Attempt to connect to the device
  } else {
      mStatus.setText(R.string.nonePaired);
      mState=getResources().getString(R.string.nonePaired);


    }


  }
```

getFirstConnectedHxm()函数将循环遍历所有已经连接的蓝牙设备，第一个以"HXM"开始的设备将被认为是我的的心率传感器设备（Zephyr），那么这个设备将会被连接。

具体代码如下：

```
private boolean getFirstConnectedHxm() {


   mHxMAddress = null;
```

```
  mHxMName = null;

  //Get the local Bluetooth adapter
 BluetoothAdapter mBtAdapter = BluetoothAdapter.getDefaultAdapter();

 /*
  *  Get a set of currently paired devices to cycle through, the Zephyr HxM must
  *  be paired to this Android device, and the bluetooth adapter must be enabled
  */
 Set<BluetoothDevice> bondedDevices = mBtAdapter.getBondedDevices();

 /*
  * For each device check to see if it starts with HXM, if it does assume it
  * is the Zephyr HxM device we want to pair with
  */
 if (bondedDevices.size() > 0) {
    for (BluetoothDevice device : bondedDevices) {
     String deviceName = device.getName();
     if ( deviceName.startsWith("HXM") ) {

     mHxMAddress = device.getAddress();
     mHxMName = device.getName();
     Log.d(TAG,"getFirstConnectedHxm() found a device whose name starts with 'HXM', its na
me is "+mHxMName+" and its address is ++mHxMAddress");
     break;
     }
    }
 }

 return (mHxMAddress != null);
 }
```

在 HxmService 类中的 connect()函数将开始一个连接的线程用于初始化一个到远程设备的连接，具体代码如下：

```
public synchronized void connect(BluetoothDevice device) {
        Log.d(TAG, "connect(): starting connection to " + device);

        // If a connection attempt is currently in progress, cancel it!
        if (mState == R.string.HXM_SERVICE_CONNECTING) {
            if (mConnectThread != null) {
                mConnectThread.cancel(); mConnectThread = null;
            }
        }
        // If a connection currently active, cancel it!
        if (mConnectedThread != null) {
            mConnectedThread.cancel();
            mConnectedThread = null;
        }

        // Make the connection
        mConnectThread = new ConnectThread(device);
        mConnectThread.start();
        setState(R.string.HXM_SERVICE_CONNECTING);
    }
```

处理 HxmService 类中发出的消息

Handler 主要用于异步消息的处理：当发出一个消息之后，首先进入一个消息队列，发送消息的函数即刻返回，而另外一个部分在消息队列中逐一将消息取出，然后对消息进行处理，也就是发送消息和接收消息不是同步的处理。在程序中 mHandler 就是用于处理 HxmService 发送给 AtyHeartRate 类的消息，具体如下代码实现：

```
private static final Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        switch(msg.what) {
            case R.string.HXM_SERVICE_MSG_STATE: {
                switch (msg.arg1) {
                    case R.string.HXM_SERVICE_CONNECTED:
                        if ((tvStatus != null) && (mHxMName != null)) {
                            tvStatus.setText(R.string.connectedTo);
                            tvStatus.append(mHxMName);
                        }
                        break;
                    case R.string.HXM_SERVICE_CONNECTING:
                        tvStatus.setText(R.string.connecting);
                        break;
                    case R.string.HXM_SERVICE_RESTING:
                        if (tvStatus != null) {
                            tvStatus.setText(R.string.notConnected);
                        }
                        break;
                }
                break;
            }
            case R.string.HXM_SERVICE_MSG_READ: {
                /**
                 * MESSAGE_READ will have the byte buffer in tow , we take it, build an instance
                 * of a HxmReading object from the bytes, and then display it into our view
                 */
                byte[] readBuf = (byte[]) msg.obj;
                hxmRead = new HxmRead(readBuf);
                R_R_interval = hxmRead.getHbTime15() - hxmRead.getHbTime14();
                updatePlotDate();
                displayRaw();
                writeToDatabase();
                break;
            }

            case R.string.HXM_SERVICE_MSG_TOAST: {
                Toast.makeText(instance.getApplicationContext(), msg.getData().getString(null), Toa
st.LENGTH_SHORT).show();
                break;
            }
        }
    }
};
```

数据的存储

传感器上传的数据是存储在 Android 手机自带的 SQLite 数据库中的。首先我们在程序中创建了一个 Heartratecontract 类，这个类主要是对在 SQL 中要用到的字符串进行初始化（包括创建表和表中相关的列名）。

具体代码如下：

```
public static final String TABLE_NAME = "heartrate";
public static final String COLUMN_ID = "entryid";
public static final String COLUMN_HEART_RATE = "heartrate";
public static final String COLUMN_SPEED = "speed";
public static final String COLUMN_DISTANCE = "distance";
public static final String COLUMN_HEART_BEAT_NUMBER = "heartbeatnumber";
public static final String COLUMN_STRIDES = "strides";

public static final String COLUMN_TIME = "timestamp";

public static final String DATABASE_CREATE = "create table "
        + Heartratecontract.TABLE_NAME + "(" + Heartratecontract.COLUMN_ID
        + " INTEGER primary key autoincrement, " + Heartratecontract.COLUMN_TIME
        + " TEXT, "+ Heartratecontract.COLUMN_HEART_RATE
```

```
            + " INT, "+ Heartratecontract.COLUMN_SPEED
            + " LONG, "+ Heartratecontract.COLUMN_DISTANCE
            + " LONG, "+ Heartratecontract.COLUMN_HEART_BEAT_NUMBER
            + " INTEGER, "+ Heartratecontract.COLUMN_STRIDES
            + " INT);";
```

在 SQLiteReaderWriter 类中，主要是更据不同的 Contract 来对数据库进行不同的操作，如数据的写入，数据的读取，

```
    /**
     * Writes given Content Values in the database
     * @param values
     * @return
     */
    public boolean writeToDatabase(ContentValues values){
         // Gets the data repository in write mode
        SQLiteDatabase db = mDbHelper.getWritableDatabase();

        System.out.println(values.toString());
        // Create a new map of values, where column names are the keys
        db.insert(mContract.getTableName(),mContract.getNullColumn(),values);

        return true;
    }

    /**
     * Returns all data that exist in the database for the specific table of the instance
     * @return
     */
    public ArrayList<HashMap<String, String>> readFromDatabase(){

         // Gets the data repository in read mode
        SQLiteDatabase db = mDbHelper.getReadableDatabase();
        String[] projection= mContract.getProjection();

        Cursor cursor = db.query(
                mContract.getTableName(),    // The table to query
                projection,                               // The columns to return
                null,                                   // The columns for the WHERE
clause
                null,                               // The values for the WHERE clause
                null,                                     // don't group the rows
                null,                                     // don't filter by row groups
                null                                    // The sort order
                );
        // Create a new map of values, where column names are the keys
        cursor.moveToFirst();
        ArrayList< HashMap<String,String> > entries = new
ArrayList<HashMap<String,String>>();

        System.out.println(mContract.getTableName()+" Records: "+cursor.getCount());
        while (!cursor.isAfterLast()) {
            HashMap<String,String> values = new HashMap<String,String>();

            for(int i=0 ; i< cursor.getColumnCount();i++){

                if(cursor.getColumnName(i).equals("timestamp")){
                    long milliSeconds= cursor.getLong(i);
                    Calendar calendar = Calendar.getInstance();
                    calendar.setTimeInMillis(milliSeconds);
                    values.put(cursor.getColumnName(i),String.valueOf(milliSeconds));
                }
                else{
                values.put(cursor.getColumnName(i),cursor.getString(i));
                }

            }

            entries.add(values);
            cursor.moveToNext();
          }
        cursor.close();
        db.close();
        return entries;

    }
```

传感器上传的相关数据的写入

在 AtyHeartRateContract 类中 writeToDatabase() 函数中主要用于将在 HxmRead 类中读取解析的数据通过上述的方式存入 SQLite 数据库中，具体代码如下：

```
private static void writeToDatabase() {
    SQLiteReaderWriter db = null;

    Contract contract = new HeartRateContract();

    if(contract != null) {
            db = new SQLiteReaderWriter(instance.getBaseContext(), contract);

            ContentValues values = new ContentValues();
            values.put(HeartRateContract.COLUMN_TIME, Calendar.getInstance().getTimeInMillis())
;
            values.put(HeartRateContract.COLUMN_DISTANCE, hxmRead.getDistance());
            values.put(HeartRateContract.COLUMN_HEART_BEAT_COUNT, hxmRead.getHeartBe
atCount());
            values.put(HeartRateContract.COLUMN_SPEED, hxmRead.getSpeed());
            values.put(HeartRateContract.COLUMN_STRIDES, hxmRead.getStrides());
            values.put(HeartRateContract.COLUMN_HEART_RATE, hxmRead.getHeartRate());

            db.writeToDatabase(values);
        }
    }
```

数据的上传

Wikihealth Android 手机 App 主要是通过 json 格式上传数据到 wikihealth 的服务器端。

首先声明一下 Url，这里的 Url 地址是 wikihealth 服务器端的地址。

```
public static final String WIKIHEALTH_RUL = "http://api2.wiki-health.org:55555/healthbook/v1
/";
```

在 AtyUserStatus 类中设置上传按钮的点击事件，在点击事件的判断逻辑中，先检查手机的网络是否可用，手机是否处于监听状态且数据库中的数据容量是否为空。在来判断是否可以进行数据的上传。具体实现如下：

```
                // Set Upload Button click listener
                btUploadData.setOnClickListener(new View.OnClickListener() {
                        DBUtils db = new DBUtils(mContext);
                        @Override
                        public void onClick(View view) {
                                // check for internet connection
                                if(!ConnectUtil.isNetWorkAvailable(mContext)) {
                                        ConnectUtil.buildAlertMessageNoInternet(mContext);
                                } else if(sIsListening) {
                                        Toast.makeText(AtyUserStatus.this, "You should stop the monitoring
process before uploading your data.", Toast.LENGTH_LONG).show();
                                } else if(db.getCapacityPercentage() == 0) {
                                        Toast.makeText(AtyUserStatus.this, "Local Database is empty. There is
nothing to upload.", Toast.LENGTH_LONG).show();
```

```
                    } else {
                        showUploadingProgress(true);

                        Toast.makeText(AtyUserStatus.this, "Uploading Data",
Toast.LENGTH_LONG).show();

                        new Thread(new Runnable() {
                            @Override
                            public void run() {
                                DataUploader dataUploader = new DataUploader(mContext);

                                try{
                                    dataUploader.uploadData();
                                } catch(Exception e) {
                                    runOnUiThread(new Runnable() {
                                        @Override
                                        public void run() {
                                            Toast.makeText(mContext, "Data uploading
interrupted due to the network problem", Toast.LENGTH_LONG).show();
                                            new UpdateUploadList().execute();

                                            showUploadingProgress(false);
                                        }
                                    });
                                }
                                runOnUiThread(new Runnable() {
                                    @Override
                                    public void run() {
                                        Toast.makeText(mContext, "Data upload
successfully", Toast.LENGTH_LONG).show();
                                        new UpdateUploadList().execute();

                                        updateDBstatus(UPLOAD_UPDATE);
                                        showUploadingProgress(false);
                                    }
                                });
                            }
                        }).start();
                    }
                }
            });

            // Set appropriate icons/labels
            updateMonitorViews();
```

```
        new UpdateUploadList().execute();
    }
```

使用 httpPost 的方式把数据封装成 json 数据的形式传递给服务器，在程序中是通过调用在 WebHttpClient 类中的 sendHttpJSONPost()函数，具体代码如下：

```
    /**
     * sends the json object to the requested URL
     */
    public static JSONObject sendHttpJSONPost (String URL, JSONObject jsonObjectSend) {
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpPost httpPostRequest = new HttpPost(URL);

        try {
            StringEntity se = new StringEntity(jsonObjectSend.toString());

            // Set parameters
            httpPostRequest.setEntity(se);
            httpPostRequest.setHeader("Content-type", "application/json;charset=UTF-8");

            HttpResponse response = httpClient.execute(httpPostRequest);

            // Get hold of the response entity (-> the data)
            HttpEntity entity = response.getEntity();

            if(entity != null) {
                // Read the content stream
                InputStream inputStream = entity.getContent();

                // convert content stream to a String
                String result = streamToString(inputStream);

                // Transform the String into a JSONObject
                JSONObject jsonObjRecv = new JSONObject(result);

                return jsonObjRecv;
            }
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (JSONException e) {
            e.printStackTrace();
        }

        return null;
    }
```

这里的 json 数据格式可以参考 wikihealth 的 API（网址：http://www.wiki-health.org/api/#!/users/createUser）。