Como usar sensores em Java ME

Neste artigo são apresentados exemplos de como usar a API de sensores do Java ME (JSR-256) em dispositivos S60 quinta edição

Visão Geral

Os dispositivos da série **S60 5th Edition** tem suporte a nova API para sensores (JSR-256). O dispositivo N97 será o primeiro dispositivo a ter essa funcionalidade como um recurso nativo (é possível instalar realizar a instalação deste recurso no outro dispositivo da série disponível, **Nokia 5800 Xpress Music**). Os dispositivos **S60 5th Edition** (e a maioria dos dispositivos da série **S60 3rd Edition FP1** e **S60 3rd Edition FP2**) tem um acelerômetro interno, que possibilita recuperar valores baseados na posição e movimento do dispositivo. Por exemplo, quando o dispositivo é colocado na posição de paisagem, a tela também é atualizada para posição de paisagem. No N97, além do acelerômetro, há três outros sensores disponíveis: sensor do nível da bateria, sensor do nível de carga e sensor da intensidade do sinal da rede.

Como apresentado em Como recuperar informações de sensores em Java ME, no Nokia N97 a API de sensores é capaz de encontrar cinco sensores diferentes. Este artigo mostra como recuperar valores do nível da bateria, sensor do estado do carregamento e sensor da intensidade do sinal da rede. O artigo How to get accelerator sensor values in Java ME (em inglês) mostra como recuperar os valores do acelerômetro em uma MIDlet.

As etapas para usar um sensor segue abaixo:

- encontre o sensor desejado no dispositivo usando o método **SensorManager.findSensors()** a partir do identificador passado como parâmetro
  - Por exemplo, em um N97 os identificadores possíveis são "acceleration", "battery_charge", "charger_state" e "network_field_intensity"
- selecione o sensor do array **SensorInfo**
  - recupere a URL para o sensor usando o método **SensorInfo.getUrl()**
- abra **SensorConnection** para o sensor
- Implemente o método **dataReceived()** para usar os dados do sensor

Examplos de código

O código abaixo mostra como buscar um sensor a partir de um identificador. Este método também recupera a URL para o sensor e retorna a instância de **SensorConnection** para o sensor selecionado.

```
    /**
     * Procura por sensores com a quantidade desejada e, se encontrada, retorna uma referência
     * SensorConnection
     * @param quantity a quantidade de sensores, por exemplo "acceleration" e "battery_charge",
     * "charger_state" and "network_field_intensity"
     * @return SensorConnection, conexão recém-aberta com o sensor selecionado com base nos critérios
     */
    private SensorConnection openSensor(String quantity) {
infos = SensorManager.findSensors(quantity, null);
if (infos.length==0) return null;
        String sensor_url = infos[0].getUrl();
try {
    return (SensorConnection)Connector.open(sensor_url);
}catch (IOException ioe) {
            ioe.printStackTrace();
    return null;
}
    }
```

O código a seguir mostra como defirir o **DataListener** para ser usado como observador dos três sensores:

```
    /**
     * Inicializa (abre) conexões com o sensor e adiciona um DataListener.
     * Cuidados também ao remover o DataListeners e fechar as conexões.
     */
    private synchronized void initSensors() {
        batterySensor = openSensor(BATTERY);
if (batterySensor == null) return;
        chargeSensor = openSensor(CHARGER);
if (chargeSensor == null) return;
        networkSensor = openSensor(NETWORK);
if (networkSensor == null) return;
        try {
    batterySensor.setDataListener(this, BUFFER_SIZE);
    chargeSensor.setDataListener(this, BUFFER_SIZE);
    networkSensor.setDataListener(this, BUFFER_SIZE);
    while(!isStopped){
        try{
     wait();
 }catch(InterruptedException ie){}
    }
    batterySensor.removeDataListener();
    chargeSensor.removeDataListener();
    networkSensor.removeDataListener();
}catch (IllegalMonitorStateException imse) {
            imse.printStackTrace();
        }catch (IllegalArgumentException iae) {
            iae.printStackTrace();
        }
try {
         batterySensor.close();
    chargeSensor.close();
    networkSensor.close();
} catch(IOException ioe){
    ioe.printStackTrace();
}
        if (isStopped) {
            batterySensor = null;
            chargeSensor = null;
```


(C) Copyright Nokia 2013. All rights reserved.

```
            networkSensor = null;
        }
    }
```

Finalmente, o método **dataReceived()** que recebe os valores do sensor. O método cria as *strings* para que seja exibido os valores dos sensores corretamente.

```
    /**
     * Notificação dos dados recebidos pelo sensor
     * @param sensor - SensorConnection, a origem do dado recebido
     * @param data - os dados recebidos pelo sensor
     * @param isDataLost - verdadeiro se algum dado foi perdido
     */
    public void dataReceived(SensorConnection sensor, Data[] data, boolean isDataLost) {
        sensorString = sensor.getSensorInfo().getQuantity();
        int values[] = data[0].getIntValues();
        if (sensorString.equals(BATTERY)) {
            batteryString = "" + values[0] + "%";
        }
        else if (sensorString.equals(CHARGER)) {
            int value = values[0];
            if (value == 0) chargeString = "not plugged in";
            else if (value == 1) chargeString = "plugged in";
        }
        else if (sensorString.equals(NETWORK)) {
            networkString = "" + values[0] + "%";
        }
        repaint();
    }
```

Os arquivos .jad e .jar também estão disponíveis aqui. A MIDlet mostra os valores dos sensores na tela, uma vez que o comando **Start** seja selecionado.

Aplicação exemplo

- SensorTest2.zip contendo **SensorTest2.jad**, **SensorTest2.jar** e os fontes.

Veja também

- Como recuperar informações de sensores em Java ME
- How to get accelerator sensor values in Java ME (em inglês)
- Mobile Sensor API (JSR-256) beta add-on for Nokia 5800 XpressMusic (em inglês)
- Nokia N97 SDK 0.5 (em inglês)
- Mobile Sensor API (JSR-256) javadoc documentation and RI Binary (em inglês)

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.