Uma aplicação utilizando Location API e Google Maps

Neste post eu irei mostrar-lhes uma aplicação interessante com a Location API. O objetivo é obter a posição do dispositivo através de GPS (com a Location API - JSR 179) e desejar a posição com o Google Maps.

A Location API (JSR 179)

A API Location (JSR 179 - pacote opcional) javax.microedition.location visa ajudar o desenvolvimento de aplicativos e serviços baseados em localização para dispositivos com recursos limitados, como telefones celulares. A aplicação deve ser definida para CLDC 2.0 (CLDC 1.0 não suporta o número de ponto flutuante, que a API utiliza para representar coordenadas e outras medidas).

A API do Google Map

Observação: Utilização deste código com a chave livre do Google Maps API infringe os termos do Google [Termos e Condições http://code.google.com/apis/maps/terms.html] (art. 10.8). Você deve comprar uma licença se você quiser utilizar a API do Google Maps, como mostrado neste exemplo.

A API do Google Maps permite-lhe incorporar o Google Maps em suas próprias páginas web com JavaScript. Portanto, vamos apenas usar um web services baseado em REST para obter a imagem do mapa e desenhar na tela do dispositivo. Nós só devemos requisitar o map através do endereço

```
http://maps.google.com/staticmap?params
```

URL e definir alguns parâmetros, tais como, latitude e longitude, tamanho da imagem, zoom, tipo de mapa (celular, etc) e, se você desejar, aplicar um marcador na posição corrente para a imagem estática. É importante notar que o exemplo a seguir foca dispositivos para a série S60.

Exemplo

A ideia básica é: (1) obter a posição do GPS e (2) passar a latitude e a longitude para os Web Services REST (URL) do Google Maps, (3) obter a imagem de resposta do servidor no formato de array de bytes. Assim, nosso primeiro passo é mostrado abaixo: O algoritmo abaixo deve ser implementada em uma thread diferente. Caso contrário, a interface do usuário pode ser bloqueada até que a resposta seja finalizada.

```
Criteria cr = new Criteria();
cr.setHorizontalAccuracy(500);
cr.setPreferredPowerConsumption(Criteria.NO_REQUIREMENT);

LocationProvider provider = null;
Location location             = null;

try {
 provider = LocationProvider.getInstance(cr);
 location = provider.getLocation(120);
 provider.setLocationListener(this, -1, 0, 0);
} catch (LocationException e) {
 alert = new Alert("Error", "Location API Error", null, AlertType.ERROR);
} catch (InterruptedException e) {
 alert = new Alert("Error", "Thread Error", null, AlertType.ERROR);
}

Coordinates c = location.getQualifiedCoordinates();
if(c != null) {
 this.lat = c.getLatitude();
 this.lon = c.getLongitude();
}
```

No início do código de exemplo acima, primeiro defini-se alguns critérios para a localização dos serviços de localização GPS. Campos de critérios incluem: precisão o tempo de resposta, a necessidade de altitude e velocidade. Estes campos de critérios irá filtrar a LocationProvider que satisfaça os valores indicados. Depois de obtido um objeto LocationProvider, o aplicativo pode usar esse objeto para obter a localização, em qualquer uma das duas maneiras:

- Invocar um método de forma síncrona para obter um único local.
- Registrar um ouvinte e obter atualizações periódicas, em intervalos definidos.

Os resultados de localização podem ser obtidos com a classe Location. Uma instância desta classe pode disponibilizar algumas informações essenciais, tais como: coordenadas, velocidade e endereço textual (se disponível), e um valor cronológico que indica quando as medições de localização foram feitas.

As coordenadas podem ser representadas em duas classes diferentes:

- Coordenadas: um ponto de latitude e longitude em graus e altitude em metros.
- QualifiedCoordinates: latitude, longitude e altitude, e também uma indicação da sua exactidão, representado como o raio de uma área.

Finalmente, temos as nossos principais informações: latitude (através do método c.getLatitude()) e longitude (através do método c.getLongitude()). Tudo o que temos de fazer agora é abrir uma conexão HTTP com o servidor do Google Maps.

```
HttpConnection httpConnection = null;
InputStream inputStream = null;
byte[] buffer = null;

try {
 httpConnection = (HttpConnection) Connector.open(url);
 httpConnection.setRequestMethod(HttpConnection.GET);
 is = httpConnection.openInputStream();
 int length = (int)httpConnection.getLength();
 if(length &gt; 0) {
  buffer = new byte[length];
  is.read(buffer);
 } else {
          int c;
  ByteArrayOutputStream byteArray = new ByteArrayOutputStream();
```

 (C) Copyright Nokia 2013. All rights reserved.

```
  while ((c = inputStream.read()) != -1) {
   byteArray.write(c);
  }
  buffer = byteArray.toByteArray();
  byteArray.close();
 }
} catch(Exception e) {
 alert = new Alert("Error", "Falha na conexão", null, AlertType.ERROR);
} finally {
 try {
  if(inputStream != null)
                  inputStream !=.close();
  if(httpConnection != null)
   httpConnection.close();
  }
 catch(Exception e2) {
  alert = new Alert("Error", "Erro de IO!", null, AlertType.ERROR);
 }
}

return buffer;
```

O código acima abre uma conexão HTTP com um determinado URL e recebe a resposta do servidor. Esta URL é um Web Service REST disponibilizado pelo Google Maps, e é como segue:

```
http://maps.google.com/staticmap?center=lat,lon&amp;zoom=16&amp;size=200x200&amp;maptype=mobile&amp;markers=lat,lon,ref/&amp;key=MAP
```

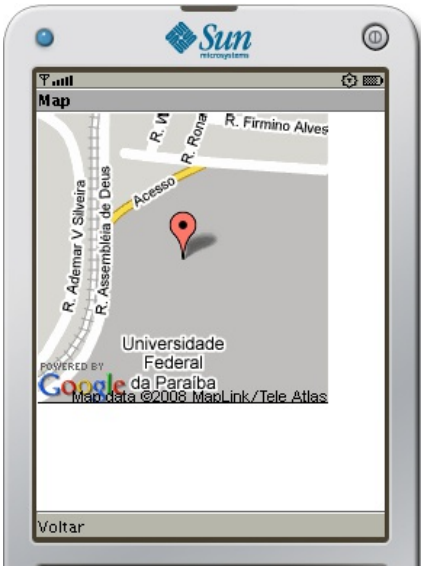
In this URL, we set some parameters for the static image: latitude and longitude, its zoom and its size (200x200), the map type (for mobile application) and a marker plotted based on lat and lon values. In our example, the application got (through GPS positioning) the Federal University of Campina Grande latitude and longitude values.

Nesta URL, define-se alguns parâmetros para a imagem estática: latitude e longitude, o seu zoom e seu tamanho (200x200), o tipo de mapa (para celular) e um marcador de localização com base em valores de lat (latitude) e lon (longitude). No nosso exemplo, a aplicação tem (através de posicionamento GPS) a latitude e a longitude da Universidade Federal de Campina Grande.

A imagem retornada pelo web service (byte[]) é utilizado para desenhar o mapa na tela.

```
Form f = new Form("Map");
f.append(Image.createImage(buffer, 0, buffer.length))
```

O resultado é:



 (C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.