Como desenvolver um jogo em Java ME - Parte 1

Então você gostaria de desenvolver para seu telefone celular? Diga alô ao Java Micro Edition. Java ME combina uma JVM pequena e um conjunto de APIs Java para desenvolver aplicações em dispositivos móveis.

Este artigo é o primeiro de uma série onde iremos percorrer todos os caminhos básicos para desenvolver uma aplicação usando Java ME. O objetivo é desenvolver um pequeno clone do jogo Arkanoid (http://en.wikipedia.org/wiki/Arkanoid) e no processo aprender todas as bibliotecas, classes e métodos disponíveis.

Desta vez, após uma rápida introdução às ferramentas de desenvolvimento, farei um guia passo-a-passo sobre como criar sua primeira aplicação Java ME, também conhecida como MIDlet.

Ferramentas de desenvolvimento

O ambiente de desenvolvimendo que você usa tem um grande impacto na sua produtividade. Após testar várias IDEs e ambientes, gostaria de focar em duas opções

- Eclipse com o plug-in EclipseME.
- Netbeans com o Mobility Pack.

Ambas as IDEs são gratuitas e possuem uma forte comunidade por trás de seu desenvolvimento. Elas permitem que você faça todo o trabalho necessário ao desenvolvimento de aplicações móveis.

Netbeans é muito fácil de instalar e configurar e sua interface é mais intuitiva do que a do Eclipse, mas ele usa mais memória e tende a ficar mais lento à medida que seus projetos crescem de tamanho.

Eclipse é mais rápido e possui um footprint de memória menor, possui as mesmas features que o Netbeans mas a interface gráfica é mais complicada de usar e portanto requer mais trabalho para configurá-lo propriamente.

Então, qual a melhor opção para seu projeto? Se você já tem experiência com o Eclipse, seu menor uso de memória e velocidade são grandes vantagens, mas se você é iniciante com IDEs para Java, o Netbeans é a melhor opção para começar.

SDKs

Além da IDE com suporte à aplicações móveis em Java, você também precisa baixar e instalar um software especial chamado SDK (Software Development Kit), que basicamente contém todas as bibliotecas Java ME e permite simulação das aplicações no próprio PC, sem necessidade de ter um dispositivo à mão.

A Sun tem um SDK chamado Wireless Toolkit (WTK) que fornece todas as bibliotecas padrão. Eu normalmente uso este SDK para meus projetos, pois ele é muito rápido e possui várias outras ferramentas úteis (Network Monitor, Profiler, SMS Emulator), além de poder ser customizados para se parecer ou agir como outros modelos de telefone.

Além do WTK também é necessário ter os SDKs dos fabricantes para os quais você deseja desenvolver sua aplicação. Estes SDKs provêm acesso a bibliotecas especiais e também incluem software que permite que você instale suas aplicações em um telefone real.

Os SDKs do Nokia Developer para o desenvolvimento de aplicações móveis podem ser encontrados no link:
http://www.developer.nokia.com/Develop/Java/Tools/

Seu primeiro midlet

Depois de instalar e configurar sua IDE Java e emuladores, é hora de começar a programar.

Todas as aplicações J2ME precisam estender a classe abstrata MIDlet, encontrada no pacote javax.microedition.midlet. O ponto de entrada da classe é o método `startApp()`.

Vamos então criar nosso primeiro MIDlet, chamado MyMidlet.

```
public class MyMidlet extends MIDlet{
  // chamado quando a aplicação é iniciada e todas as vezes em que ela é continuada
  // depois do pauseApp
  protected void startApp() throws MIDletStateChangeException {}

  // chamado quando o MIDlet está para ser destruído
  protected void destroyApp(boolean uncondicional) throws MIDletStateChangeException {}

  // chamado quando o MIDlet está para ser pausado
  protected void pauseApp() {}
}
```

Se você criar esta classe, compilá-la e rodá-la no emulador, você terá uma fantástica tela branca!

Para mostrar conteúdo em sua aplicação você precisa usar a class `Display`. Esta classe controla o que aparece na tela de um MIDlet. Cada MIDlet tem um Display e pode acessá-lo usando o método estático `Display.getDisplay(this)`

(C) Copyright Nokia 2013. All rights reserved.

Para mostrar algo na tela você precisa definir um objeto derivado de `Displayable`, usando o método `setCurrent()`. Uma das classes que implementam a interface Displayable é a classe `Alert`, que mostra uma mensagem na tela.

Com esta informação, vamos mudar nosso método startApp() para que ele faça algo mais útil:

```
Alert alert;

  //ponto de entrada
protected void startApp() throws MIDletStateChangeException {

  //cria um alerta
  alert = new Alert("Hello World");

  //mostra o alerta na tela.
  Display.getDisplay(this).setCurrent(alert);
}
```

Rodando seu midlet no emulador WTK, você terá o resultado mostrado na imagem abaixo:

Isso é um pouco melhor, mas você ainda não tem uma maneira de sair do seu MIDlet corretamente. Adicionando as seguintes linhas de código ao método startApp() cria um comando que permitirá a saída da aplicação:

```
// cria comando
comExit = new Command("Exit", Command.EXIT, 1);
// adiciona um comando para sair do Midlet
mainForm.addCommand(comExit);
```

Isto mostra o comando "Exit" na tela, mas se você clicar sobre ele, nada acontecerá, pois você ainda não adicionou o código necessário para reagir a comandos em seu MIDlet. Para tal, você precisa adicionar uma instância da classe `CommandListener` ao seu alerta, de modo que você possa tratar os comandos recebidos e seguir com as ações necessárias.

```
public class MyMidlet extends MIDlet implements CommandListener{

  public void startApp(){
    [...]
    // adiciona um listener de comandos ao alerta
    alert.setCommandListener(this);
  }

  public void commandAction(Command cmd, Displayable display) {
    // checa qual comando foi acionado
    if (cmd == comExit) {
      notifyDestroyed();
    }
  }
}
```

Com esta mudança finalmente temos um MIDlet completo e funcional.

Na próxima lição entraremos em mais detalhes sobre a interface gráfica e seus elementos, como Alert, Display e CommandListener. Usaremos esses elementos para definir um menu gráfico para nosso clone do Arkanoid.

Downloads:

- Código-fonte
- Jad File
- Jar File

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.