



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

(VENDA E DISTRIBUIÇÃO PROIBIDA)

De : Luciano Alves da Silva (lucianopascal@yahoo.com.br)

www.apostilaandroid.net

**Rio de Janeiro
Dezembro 2012**



Aviso sobre esta apostila

Antes de iniciar a leitura deste material, veja esses avisos importantes:

Esse material **NÃO PODERÁ SER DISTRIBUÍDO**, em hipótese alguma, em outros sites da Internet ou através outros processos/meios .

Esse material , em hipótese alguma, **NÃO PODE SER COMERCIALIZADO** tanto pela Internet ou de forma impressa.

Se por acaso você encontrar este material sendo distribuído em outro site ou sendo comercializado (sem ser pelo site oficial da apostila), por favor, entre em contato com o autor (ver e-mail na primeira página).



Sobre o Autor da Apostila

Luciano Alves da Silva é Bacharelado em Ciência da Computação pela UNISUAM e Pós-Graduado em Docência do Ensino Superior pelo Instituto A Vez do Mestre (Universidade Cândido Mendes - UCAM). Possui conhecimento e domínio das linguagens de programação Pascal, Java, C/C++, C#, Visual Basic, Delphi, PHP e HTML. Já criou Ambientes de Desenvolvimento Integrado (conhecidos como IDE) como o MakeWare (que trabalha com as linguagens Pascal, C++ e Java) e o AlgoWare (interpretador de algoritmos).

É autor também dos seguintes livros, pela editora AGBOOK

- Aprenda Passo a Passo a Programar em Android – Guia Essencial para Desenvolvedores
- Desenvolvendo Jogos com a Plataforma XNA – Guia para Desenvolvedores.
- Desenvolvendo Jogos com a Ferramenta RPG Maker VX– Guia do Usuário.



Apresentação

Este material é dedicado para aqueles que desejam construir aplicações que façam uso da Internet no Android.

Caso você, que está lendo este material, seja iniciante em programação com Android, recomendo antes adquirir no site oficial a “Apostila de Android – Programando Passo a Passo 5ª Edição”, pois esse material nada mais é do que um “complemento” da apostila citada. Agora caso você já possua conhecimento básico sobre programação com Android, e deseja aprender a construir aplicações que façam uso da Internet, este material lhe dará todo o passo a passo para a construção de programas em Android que trabalhem com Internet.



Índice analítico

Capítulo 1 Programação com Internet no Android	6
1.1) Trabalhando com sockets em Android	6
1.2) Desenvolvendo uma aplicação cliente em Android e um servidor Java...	7
1.3) Obtendo conteúdo de uma URL	20
1.4) Enviando informações para uma página Web via URL	25
1.5) Enviando informações para uma página Web método POST	31
1.5.1) Diferenças entre GET e POST	31
1.5.2) Desenvolvendo a aplicação de envio de dados (usando método POST)	32
1.6) Fazendo download de um arquivo	38
1.7) O componente WebView	42
1.8) Desenvolvendo uma aplicação que visita páginas Web	44
Conclusão a respeito do material	46



Capítulo 1 Programação com Internet no Android

Atualmente as aplicações Android (digamos que praticamente todas elas) de hoje em dia, de alguma forma, fazem uso da Internet (seja para obter informações, enviar dados e etc.). Neste capítulo iremos aprender a desenvolver aplicações que farão uso da Internet, através dos exemplos contidos neste capítulo.

Antes de começarmos a começar a programar no Android , precisamos entender como funciona todo esse processo de comunicação das aplicações e/ou serviços através da Internet.

1.1) Trabalhando com sockets em Android

O que são sockets ou soquetes ?

Os sockets são conexões de internet que são especificadas pelos protocolos TCP/IP, ou melhor, são endereços de “nós” (um dispositivo como um computador e etc.) e um número de porta, os quais identificam um serviço.

Como funciona ?

- 1) O programa servidor começa a ser executado esperando por um pedido do cliente.
- 2) O programa cliente requisita uma conexão indicando o servidor com o qual quer se conectar.



3) Quando o cliente envia um pedido, o servidor pode “aceitar” a conexão, e inicia um “soquete” no lado servidor e dedica esse soquete para essa conexão específica com o cliente.

Todo computador ou dispositivo conectado a internet pode ser acessado por qualquer outro computador ou dispositivo, e eles podem estar atuando como cliente (que requisita uma conexão) ou servidor (negando ou aceitando essa conexão).

A idéia aqui é criar um cliente, que será uma aplicação em Android onde ela irá requisitar um serviço no servidor (que será uma aplicação “Desktop” que iremos desenvolver em Java). A aplicação cliente será uma calculadora onde ela envia os dois números para o servidor, que fará o cálculo e posteriormente retornará a soma dos dois números para o cliente. Existem duas classes Java utilizadas aqui para trabalhar com sockets que são **ServerSocket** (usada no lado servidor) e **Socket** (usada no lado cliente).

1.2) Desenvolvendo uma aplicação cliente em Android e um servidor Java.

A classe **ServerSocket** é utilizada para “escutar” as conexões clientes que vem da rede. Bom, vamos criar uma aplicação em Java que será o nosso “servidor”, que irá receber as requisições da aplicação cliente, que será desenvolvida em Android.

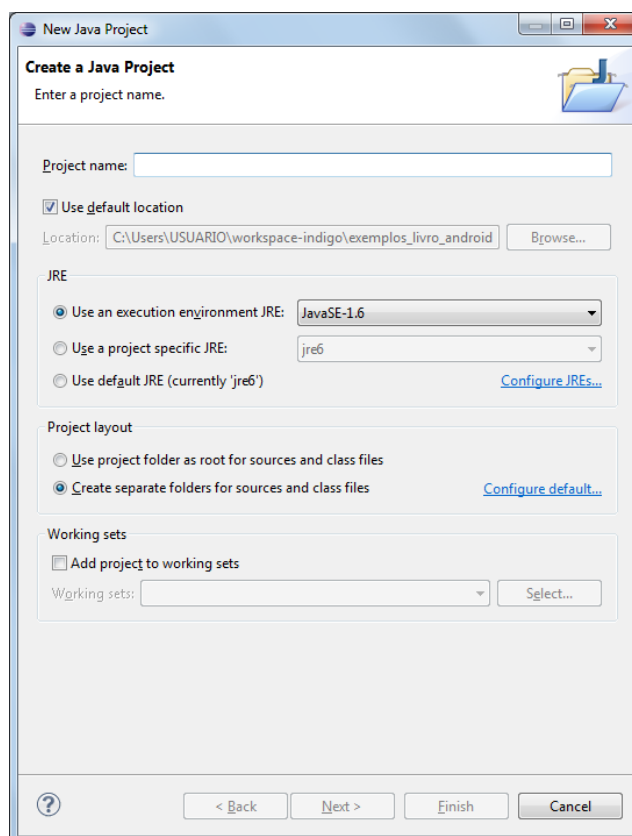
Para criarmos uma aplicação Java (no estilo “Desktop”) vamos no menu “File”/”New”/”Java Project”. Feito isso será aberta uma caixa de diálogo conforme demonstra a figura seguinte:



Apostila de Android

Programando Passo a Passo

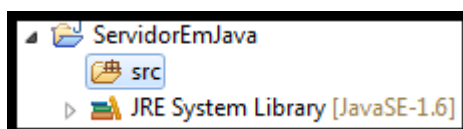
Programação com Internet (Edição Completa)



Caixa de diálogo – New Java Project

No campo “Project Name” vamos digitar o nome do nosso projeto que vai se chamar “ServidorEmJava”, logo após clique no botão “Finish” para que o projeto possa ser criado.

Expanda o item “ServidorEmJava” e selecione a pasta “src”, conforme a figura abaixo:



Pasta “ServidorEmJava” expandida

Agora vamos criar uma nova classe Java, simplesmente clicando com o botão direito sobre a pasta “src” e em seguida selecionando “New”/“Class”.

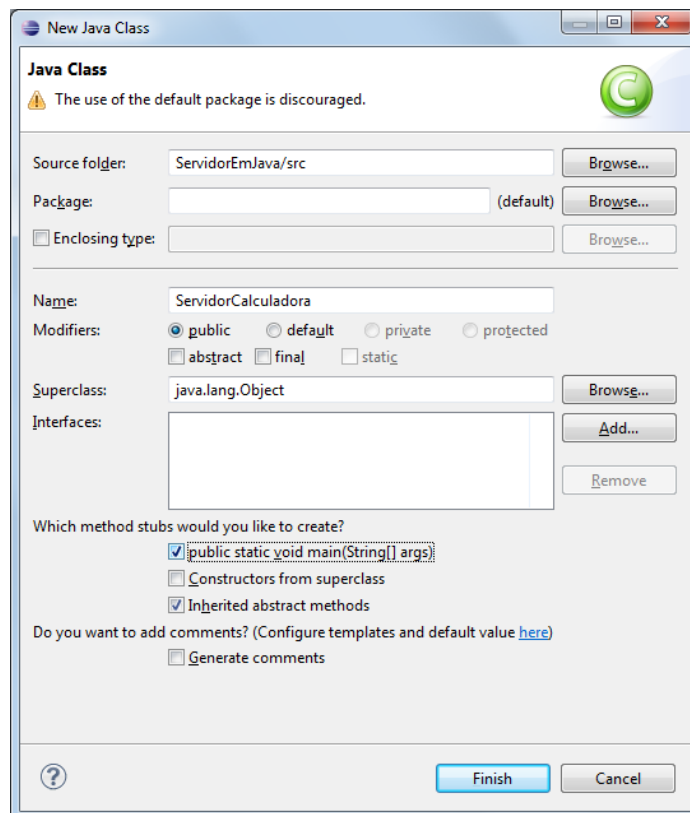


Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

No campo name digite “ServidorCalculadora” e marque a opção “public static void main(String arg[])”, conforme a figura seguinte:



Criando uma nova classe

Depois de preencher os campos acima clique em “Finish” para criar a classe. Depois disso vamos digitar o código abaixo dentro da classe:

```
import java.net.*;
import java.io.*;

public class ServidorCalculadora {

    public static void main(String[] args) {

        double num1, num2;

        try {
            ServerSocket serverSocket = new ServerSocket(60);
            System.out.print("Aguardando conexao...\n\n");
            Socket socket = serverSocket.accept();
            System.out.print("Conexao aceita...\n\n");
```



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

```
DataInputStream in = new
DataInputStream(socket.getInputStream());

DataOutputStream out = new
DataOutputStream(socket.getOutputStream());

num1 = in.readDouble();
num2 = in.readDouble();

System.out.println(num1 + "+" + num2 + "=" +
(num1 + num2));

out.writeDouble((num1 + num2));
in.close();
out.close();
socket.close();
serverSocket.close();

} catch (IOException e) {
    e.printStackTrace();
}

}
}
```

Vamos ao comentário de algumas linhas de código. A linha :

```
ServerSocket serverSocket = new ServerSocket(60);
```

Cria uma instância na classe **ServerSocket**, passando como parâmetro do construtor o número da porta pela qual o socket irá escutar as conexões clientes, que será 60. Na linha:

```
Socket socket = ss.accept();
```

Acontece a seguinte situação: o objeto “*serverSocket*” chama o método **accept**, que é responsável por aguardar uma conexão cliente que virá na porta 60. Enquanto nenhuma conexão vir, essa linha ficará “pendurada”, ou seja, ficará aguardando uma conexão cliente, até ela acontecer. Quando vier uma conexão, a referência do cliente será armazenada na variável “*socket*”, do tipo **Socket**. Nas instruções abaixo :



```
DataInputStream in  = new DataInputStream(socket.getInputStream());  
DataOutputStream out = new DataOutputStream(socket.getOutputStream());
```

São criados dois objetos. O primeiro objeto se chama *in*, do tipo **DataInputStream**, que é uma classe responsável por receber dados que vem de uma conexão. O segundo objeto se chama *out*, do tipo **DataOutputStream**, que é uma classe responsável por enviar dados para uma conexão.

Nas instruções :

```
num1 = in.readDouble();  
num2 = in.readDouble();
```

Ocorre o seguinte : as variáveis *num1* e *num2* recebem os números que são enviados pelo cliente, pelo método **readDouble**.

A instrução:

```
out.writeDouble((num1 + num2));
```

Envia para a aplicação cliente, a soma dos dois números, pelo método **writeDouble**. As instruções:

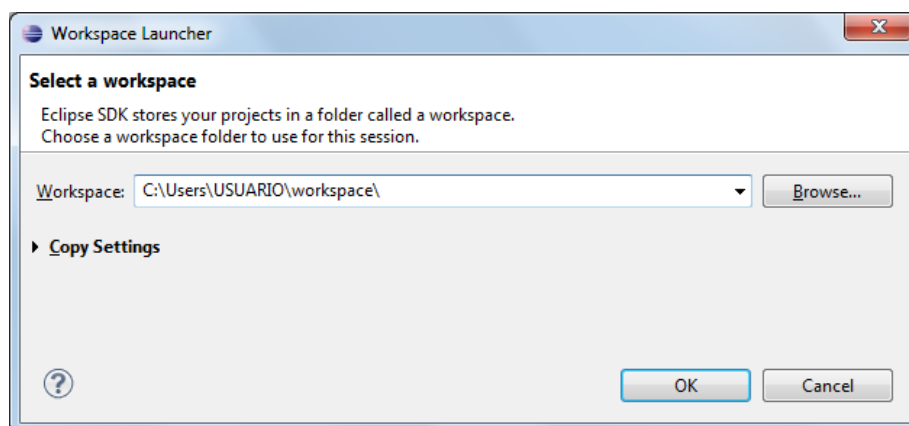
```
in.close();  
out.close();  
socket.close();  
serverSocket.close();
```

Fecha todas as conexões.



Agora salve o arquivo, mas, não o execute. Iremos executar esse arquivo fora do Eclipse. Quando salvamos um arquivo, e caso a sintaxe dele estiver correta, o Eclipse “automaticamente” compila o arquivo, gerando o seu executável (o arquivo “.class”). Agora precisamos executar o nosso servidor, mas, primeiramente precisamos saber onde esse executável se encontra ? Quando executamos o Eclipse pela primeira vez ele pede um diretório para trabalho (workspace), e automaticamente ele já sugere um, mas você pode mudá-lo (caso precise).

Agora, você lembra qual é o diretório que você selecionou para workspace no Eclipse ? Se você não se lembra, faça o seguinte : Vá no menu “File”/”Switch Workspace”/”Other...”, e será mostrada a caixa de diálogo abaixo:



Diretório do workspace

Agora se lembrou né ? Simplesmente cancele a caixa de diálogo mostrada na figura acima e vá no diretório mostrado na caixa de diálogo acima (é lá que se encontra o nosso projeto, juntamente com seu executável).

Dentro do diretório “ServidorEmJava” existe um diretório chamado “bin”, onde estão armazenados todos os executáveis do Java (arquivos “.class”).

Agora, vá no “prompt” do comando e vamos executar essa classe, como ? A minha classe se encontra no seguinte caminho:

```
"C:\Users\USUARIO\workspace\ServidorEmJava\bin"
```

Quando eu for executar a minha classe, será assim:

```
java ServidorCalculadora
```



Veja um exemplo na figura abaixo:

```
C:\Windows\system32\cmd.exe - java ServidorCalculadora
C:\Users\USUARIO\workspace\ServidorEmJava\bin>java ServidorCalculadora
Aguardando conexao...
```

Aplicação servidor em execução

Você vai “minimizar” essa janela, deixando o servidor em execução. Agora vamos criar a nossa aplicação cliente em Android, conforme mostrada os dados abaixo:

Application Name: AplicacaoClienteAndroid

Project Name: AplicacaoClienteAndroid

Package Name : com.example.aplicacaoclienteandroid

Build SDK : Android 2.2 (API 8)

Minimum Required SDK : API 8: Android 2.2 (Froyo)

Activity Name: AplicacaoClienteActivity

Layout Name : activity_aplicacao_cliente

Title : Aplicação Cliente

Após o projeto ser criado, copie e cole o código XML abaixo no arquivo “activity_aplicacao_cliente.xml” :



```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Digite o primeiro número" />
    <EditText
        android:id="@+id/ednumero1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberSigned" >

        <requestFocus />
    </EditText>

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Digite o segundo número" />

    <EditText
        android:id="@+id/ednumero2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="number" />

    <Button
        android:id="@+id/btobtersoma"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Obter soma" />

</LinearLayout>
```

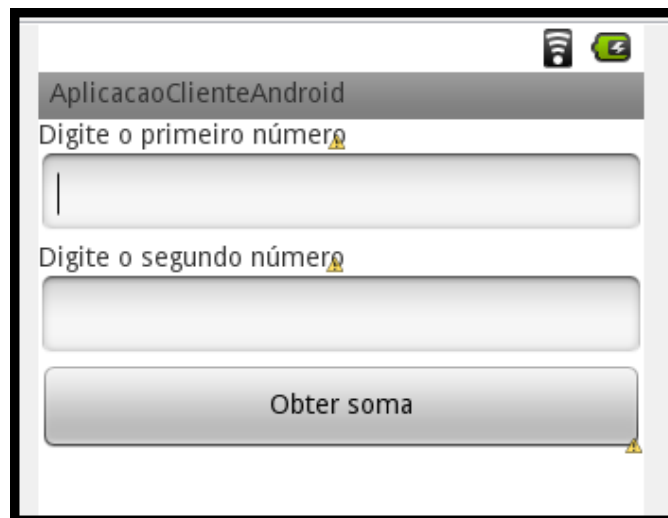
A cara da aplicação deve estar de acordo como mostra a figura seguinte:



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)



Layout da tela da aplicação

Agora vamos no arquivo “AplicacaoClienteActivity.java” para digitarmos o seguinte código abaixo:

```
package com.example.aplicacaoclienteandroid;

import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
import android.view.View;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.*;

import android.widget.*;
import android.view.*;

public class AplicacaoClienteActivity extends Activity {

    Socket socket;

    EditText ednumero1, ednumero2;

    Button btobtersoma;

    DataInputStream in;

    DataOutputStream out;

    @Override
    public void onCreate(Bundle savedInstanceState) {
```



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

```
super.onCreate(savedInstanceState);

setContentView(R.layout.
    activity_aplicacao_cliente);

ednumero1 = (EditText) findViewById
    (R.id.ednumero1);

ednumero2 = (EditText) findViewById
    (R.id.ednumero2);

btobtersoma = (Button) findViewById
    (R.id.btobtersoma);

try {

    socket = new Socket("10.0.2.2", 60);

    in = new DataInputStream
        (socket.getInputStream());

    out = new DataOutputStream
        (socket.getOutputStream());

    btobtersoma.setOnClickListener(new
    View.OnClickListener() {

        @Override
        public void onClick(View arg0) {

            try
            {
                out.writeDouble(Double.parseDouble
                (ednumero1.getText().toString()));
                out.writeDouble(Double.parseDouble
                (ednumero2.getText().toString()));

                double resultado_soma = in.readDouble();

                ExibirMensagem("A soma é : " + resultado_soma);

            }catch (Exception e) {

                ExibirMensagem("Erro ao obter soma");
            }

        }
    });
}
```




```
}catch (Exception e) {  
  
    ExibirMensagem("Erro ao efetuar conexão");  
  
}  
  
  
public void ExibirMensagem(String mensagem){  
  
    AlertDialog.Builder dialogo = new AlertDialog.  
        Builder(AplicacaoClienteActivity.this);  
  
    dialogo.setTitle("Mensagem");  
    dialogo.setMessage(mensagem);  
    dialogo.setNeutralButton("OK", null);  
    dialogo.show();  
}  
  
}
```

Irei explicar agora algumas partes de código desse programa. A linha:

```
socket = new Socket("10.0.2.2",60);
```

Cria uma instância da classe **Socket**, passando como parâmetros o endereço IP do nosso servidor e a porta pela qual nós iremos se comunicar. Espere, o endereço IP local normalmente não é 127.0.0.1 ou localhost? Sim, mas se passássemos como parâmetro o IP 127.0.0.1 ou localhost nós estaríamos se comunicando com o emulador e não com o computador que está com o servidor em execução. No Android, todo endereço IP local de um computador é referenciado pelo IP 10.0.2.2, que é um endereço local de rede.

As instruções :

```
out.writeDouble(Double.parseDouble(ednumero1.  
getText().toString()));  
  
out.writeDouble(Double.parseDouble(ednumero2.  
getText().toString()));
```



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

Enviam para o servidor, os números a serem somados, pelo método **writeDouble**. A instrução :

```
double resultado_soma = in.readDouble();  
  
ExibirMensagem("A soma é : " + resultado_soma);
```

Chama o método **readDouble**, que retorna a soma que será enviada pelo servidor para ser guardada na variável *resultado_soma*.

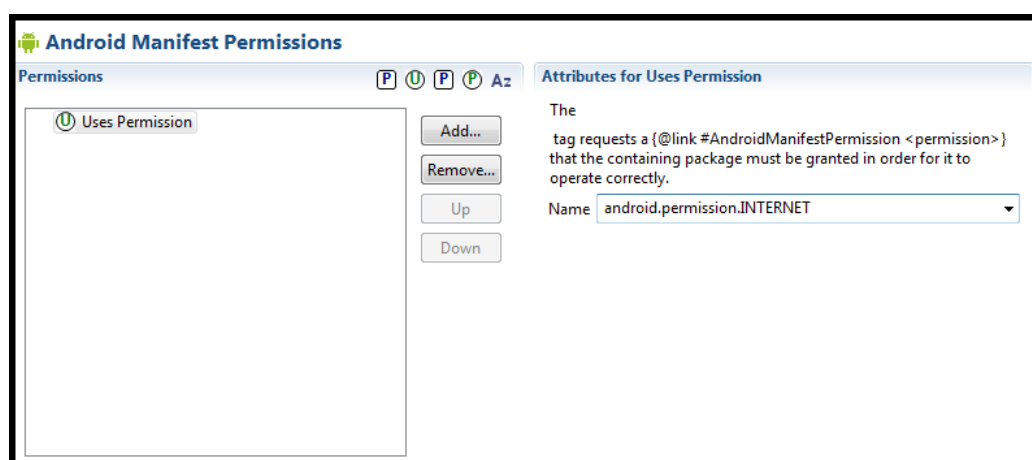
Em seguida a soma é exibida na tela através do método **ExibirMensagem**.

Se executarmos esse programa vamos ter um erro de conexão, pois, para acessar a Internet a aplicação precisa de uma “permissão”, logo, precisamos definir uma permissão liberando a nossa aplicação de usar a Internet (como nós fizemos para realizar uma chamada telefônica).

Para definirmos uma permissão vamos no arquivo “AndroidManifest.xml” para carregá-lo e em seguida vamos na seção “Permissions”. Nesta seção vamos clicar no botão “Add” para adicionarmos a permissão (Uses Permission) que vai liberar a nossa aplicação para usar a Internet. A permissão que permite isso é:

```
android.permission.INTERNET
```

Veja como ficou configurado na figura seguinte:



Definindo a permissão de acesso a Internet



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

Vamos executar a nossa aplicação. O resultado você vê na figura abaixo:



Aplicação cliente Android em execução

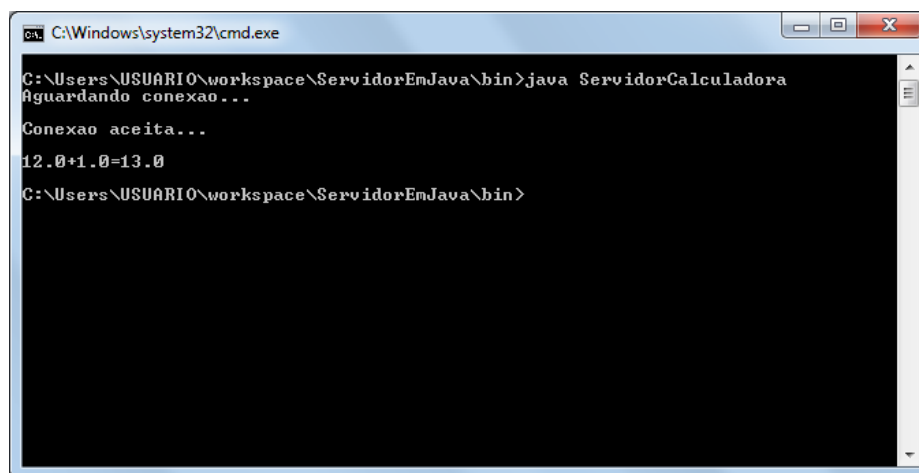
Experimente digitar dois números quaisquer e depois, clique no botão “Somar”. Será exibida uma mensagem com o valor da soma, como mostra um exemplo da figura abaixo:



Aplicação cliente Android em execução



Lembra-se que eu pedi para que você deixasse minimizado aquela janela de prompt, pela qual executamos o servidor? Maximize-a e veja o que aconteceu, como mostra a figura seguinte:



```
C:\Windows\system32\cmd.exe
C:\Users\USUARIO\workspace\ServidorEmJava\bin>java ServidorCalculadora
Aguardando conexao...
Conexao aceita...
12.0+1.0=13.0
C:\Users\USUARIO\workspace\ServidorEmJava\bin>
```

Aplicação servidor em execução: Resultado

O servidor aceitou a conexão, recebeu os dois números e os enviou de volta para o cliente Android.

Observação importante: O programa acima lê somente os valores enviados pelo cliente (aplicação Android) apenas uma única vez. Depois que a soma é retornada para o cliente a aplicação servidor encerra sua execução, logo, você terá que executar novamente o servidor e depois novamente terá também que executar a sua aplicação cliente Android.

1.3) Obtendo conteúdo de uma URL

Agora irei mostrar neste tópico como desenvolver uma aplicação que obtém um conteúdo de uma determinada URL (endereço de alguma página ou conteúdo web), que será especificada através da aplicação.

Crie um novo projeto com os seguintes dados abaixo:

Application Name : ObtendoConteudoURL



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

Project Name: ObtendoConteudoURL

Package Name : com.example.obtendoconteudourl

Build SDK : Android 2.2 (API 8)

Minimum Required SDK : API 8: Android 2.2 (Froyo)

Activity Name: ConteudoURLActivity

Layout Name : activity_conteudo_url

Title : Obtendo Conteúdo da URL

Após carregar o projeto altere a estrutura da tela para **LinearLayout** (Vertical) e em seguida altere o título do componente **TextView** conforme é mostrado em seguida:

TextView

Propriedade	Valor
Text	Digite aqui sua URL:

Depois de fazer o que foi pedido acima, insira os seguintes componentes na sequência:

EditText

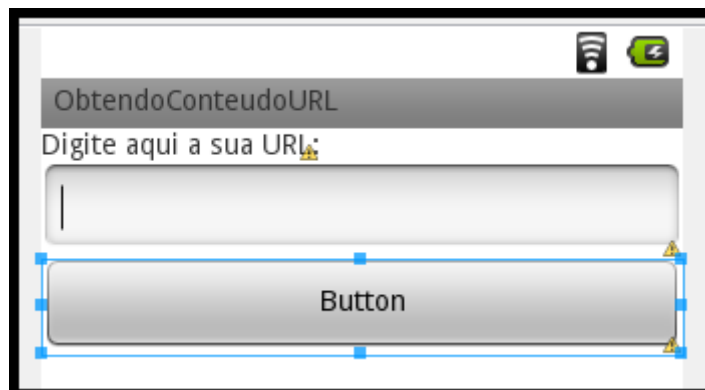
Propriedade	Valor
Id	edurl
Width	fill_parent

Button

Propriedade	Valor
Id	btmostrar_conteudo
Width	fill_parent
Text	Mostrar



Seguindo os passos acima, o layout da aplicação deve estar de acordo com a figura abaixo:



Layout da tela da aplicação

Agora vamos no arquivo “ConteudoURLActivity.java” para digitarmos o seguinte código:

```
package com.example.obtendoconteudourl;

import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
import android.widget.*;
import android.view.*;
import java.io.*;
import java.net.*;

public class ConteudoURLActivity extends Activity {

    EditText edurl;

    Button btmostrar_conteudo;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_conteudo_url);

        edurl = (EditText) findViewById(R.id.edurl);
        btmostrar_conteudo = (Button) findViewById(
            R.id.btmostrar_conteudo);

        btmostrar_conteudo.setOnClickListener(new View.
            OnClickListener() {
```



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

```
@Override
public void onClick(View arg0) {

    try {
        String strurl =
            edurl.getText().toString();

        URL url = new URL(strurl);

        InputStream is = url.openStream();

        int i;

        String conteudo="";

        while((i = is.read()) != -1)
            conteudo += ((char)i);

        ExibirMensagem(conteudo);

    } catch (Exception e) {

        ExibirMensagem("Erro ao obter conteúdo.");

    }
}

public void ExibirMensagem(String mensagem){

    AlertDialog.Builder dialogo = new AlertDialog.
        Builder(ConteudoURLActivity.this);
    dialogo.setTitle("Aviso");
    dialogo.setMessage(mensagem);
    dialogo.setNeutralButton("OK", null);
    dialogo.show();

}

}
```

Nesse programa que desenvolvemos existe um campo (do tipo **EditText**) através do qual irei digitar uma URL de minha escolha para obter o seu conteúdo (que pode ser de página Web ou um arquivo de texto simples) e um botão que será responsável por “obter” o conteúdo dessa URL. Vamos analisar o código do evento desse botão responsável por obter o conteúdo da URL.



A linha:

```
String strurl = edurl.getText().toString();
```

Cria uma variável do tipo **String** chamada *strurl* , que vai conter a URL digitada pelo usuário.

A linha:

```
URL url = new URL(strurl);
```

Cria uma instância da classe **URL**, responsável por acessar uma determinada URL. Neste caso, a URL está contida dentro da variável *strurl*.

A linha :

```
InputStream is = url.openStream();
```

Retorna para a variável chamada *is*, do tipo **InputStream**, o fluxo obtido pela URL, através do método **openStream**. O código:

```
while((i=is.read())!= -1)
    s+=(char) i;
```

Joga para a variável *s*, do tipo **String**, o conteúdo da URL.

Só para lembrar que para executar essa aplicação, é necessário definir a permissão de acesso a Internet, como nós fizemos na aplicação anterior.

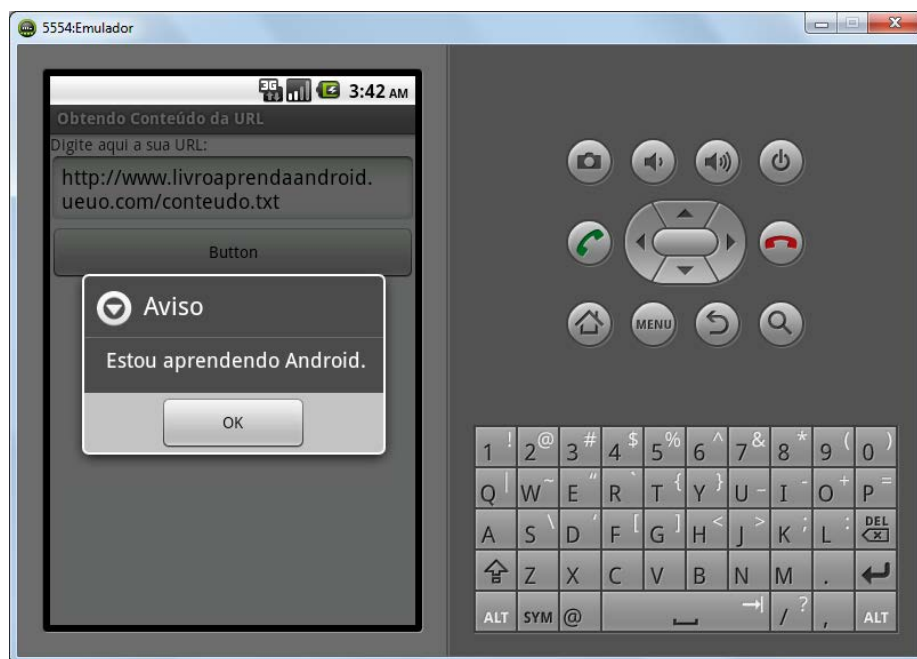
Execute a aplicação. Veja o resultado de sua execução abaixo:



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)



Aplicação em execução

Como havia falado, você pode experimentar uma outra URL, porém, é muito provável que páginas muito grandes façam com que a aplicação não responda ou talvez você se depare com um erro de execução, pois, um conteúdo muito grande de uma URL não seja talvez “comportado” pela caixa de diálogo.

1.4) Enviando informações para uma página Web via URL

Utilizando os mesmos conceitos do programa anterior, vamos construir uma aplicação Android que envia informações via URL para uma determinada página Web.

Para esse exemplo, desenvolvi uma página PHP no site do livro do Android (<http://www.livroaprendaandroid.ueuo.com>) para que ele possa receber os dados enviados por nossa aplicação. A aplicação consiste em um cadastro onde os dados serão armazenados em uma página Web, e esses dados serão passados via URL para o site.

Crie um novo projeto com os seguintes dados abaixo:

Application Name: EnviandoDadosURL

Project Name: EnviandoDadosURL

Package Name : com.example.enviandodadosurl



Build SDK : Android 2.2 (API 8)

Minimum Required SDK : API 8: Android 2.2 (Froyo)

Activity Name: EnviandoDadosActivity

Layout Name : activity_enviando_dados

Title : Enviando Dados via URL

Após criar o projeto, altere a estrutura da tela para **LinearLayout** (Vertical) em seguida altere a propriedade do componente **TextView** , conforme abaixo:

TextView

Propriedade	Valor
Text	Nome:
Padding	(Deixar em branco)

Em seguida adicione os seguintes componentes na sequência:

EditText

Propriedade	Valor
Id	ednome

TextView

Propriedade	Valor
Text	Idade:

EditText

Propriedade	Valor
Id	edidade



Button

Propriedade	Valor
Id	btenviar_dados
Width	fill_parent
Text	Enviar dados

Seguido os passos acima, a tela da nossa aplicação deve estar de acordo com a figura abaixo:



Layout da tela da aplicação

Agora no arquivo “EnviandoDadosActivity.java” vamos digitar o seguinte código abaixo:

```
package com.example.enviandodadosurl;

import android.os.Bundle;
import android.app.*;
import android.widget.*;
import android.view.*;

import java.net.*;

public class EnviandoDadosActivity extends Activity {

    EditText ednome, edidade;
```



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

```
Button btenviar_dados;
```

```
int id;
```

```
@Override
```

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.  
        activity_enviando_dados);
```

```
    id = (int) Math.round(Math.random() * 10000)  
        + 10000;
```

```
    ednome = (EditText) findViewById(R.id.ednome);  
    edidade = (EditText) findViewById(R.id.edidade);
```

```
    btenviar_dados = (Button) findViewById  
        (R.id.btenviar_dados);
```

```
    btenviar_dados.setOnClickListener(new View.  
        OnClickListener() {
```

```
        @Override
```

```
        public void onClick(View arg0) {
```

```
            String nome = URLEncoder.encode  
                (ednome.getText().toString());
```

```
            String idade = edidade.getText().  
                toString();
```

```
        try {
```

```
            URL url = new URL("http://www.  
                livroaprendaandroid.ueuo.com/  
                cadastrar_dados_url.php?id=  
                " + id + "&nome=" + nome + "&idade=" +  
                idade);
```

```
            url.openStream();
```

```
            ExibirMensagem("Dados enviados com sucesso.  
                Seu ID para consulta é : " + id);
```

```
        } catch (Exception e) {
```

```
            ExibirMensagem("Erro ao enviar mensagem");  
        }
```



```
        }  
    });  
}  
  
public void ExibirMensagem(String mensagem){  
  
    AlertDialog.Builder dialogo = new AlertDialog.  
        Builder(EnviandoDadosActivity.this);  
    dialogo.setTitle("Aviso");  
    dialogo.setMessage(mensagem);  
    dialogo.setNeutralButton("OK", null);  
    dialogo.show();  
}  
}
```

O código dessa aplicação é similar ao da aplicação anterior, então não há muitos comentários a ser fazer dessa aplicação, porém, há algumas linhas de código que preciso explicar.

Observe que passamos os parâmetros “id”, “nome” e “idade” para a URL certo ? A idade é um valor numérico simples , já o nome é um conjunto formado por vários caracteres comuns e especiais. Quando passamos parâmetros para uma URL, nessa URL não pode haver caracteres especiais , para isso fiz uso do método **encode** da classe **URLEncoder**, responsável por converter esses caracteres especiais em seus respectivos valores hexadecimais, que são aceitos pela URL, conforme mostro abaixo:

```
nome = URLEncoder.encode(nome);
```

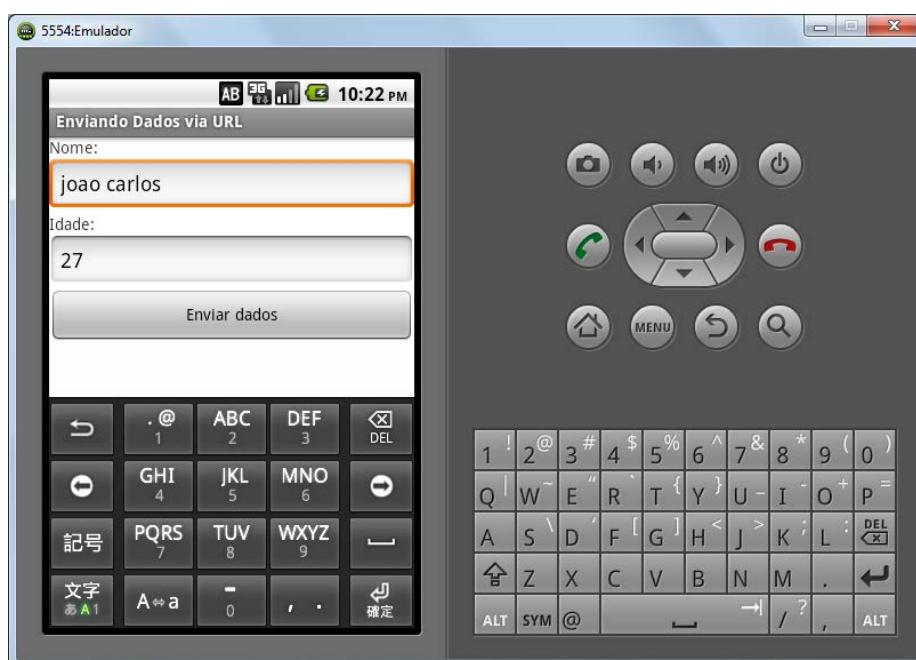
Antes de executar a aplicação, lembre-se de que você deve configurar a “permissão” de acesso a Internet no Android. Feito isso você já pode executar a aplicação, conforme demonstra a figura seguinte:



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)



Aplicação de envio de dados em execução

Quando clicamos no botão “Enviar dados”, os dados são enviados e na mensagem obtemos a ID da mensagem, veja na figura seguinte:



Dados enviados com sucesso

Para fazer a consulta (para confirmar se os dados foram enviados), use o link :

http://livroaprendaandroid.ueuo.com/consultar_dados_url.php?id=<id da consulta>



Apostila de Android

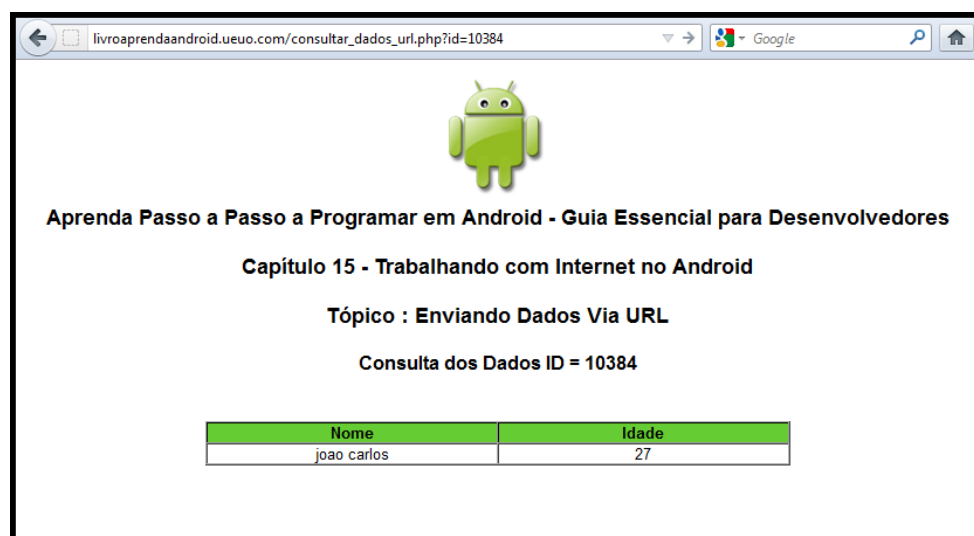
Programando Passo a Passo

Programação com Internet (Edição Completa)

Vimos que no exemplo acima, a ID passada foi 10384, logo, para consultar precisamos digitar o seguinte link no navegador:

http://livroaprendaandroid.ueuo.com/consultar_dados_url.php?id=10384

O resultado você confere na figura seguinte:



Conferindo o resultado do envio dos dados

1.5) Enviando informações para uma página Web método POST

Desenvolvemos uma aplicação que enviava dados para o servidor através de sua URL (método GET). Agora vamos desenvolver um aplicação similar à aplicação anterior, só que agora essa aplicação fará uso do método POST, a através do uso da classe **HttpPost**, do Android.

1.5.1) Diferenças entre GET e POST

Para alguns programadores, os métodos GET e POST são métodos que executam ações diferentes uma das outras, o que não é verdade. Os métodos GET e POST tem a mesma finalidade, que é o de enviar dados para uma página Web. A diferença está no “método de envio” deles. Vamos citá-los:

O método GET envia dados para o servidor utilizando como meio de trafego desses dados a URL, de onde se encontra o servidor. Por exemplo: quero



enviar para meu servidor dois números para serem processados por uma página Web. Pelo GET faríamos assim:

<http://www.meuservidor.com/calculadora.php?numero=12&numero2=13>

Observe que na URL acima existe uma página que vai processar os números que quero mandar pra ela, no caso, a página “calculadora.php”. Observe que nessa página são passados dois parâmetros : “numero”, que recebe como valor 12 e “numero2”, que recebe como valor 13. Esses dois parâmetros são reconhecidos pela página “calculadora.php”, que fará o processamento deles.

Uma das características do GET é que nesse método de envio, podemos “visualizar” os dados que são enviados para a pagina no servidor, o que , em algumas ocasiões, não é seguro. Vamos supor que você enviasse via GET dados de nome e senha, facilmente esses dados seriam de alguma forma “capturados” e usados de forma indevida.

Uma outra característica do método GET está relacionado à limitação de dados a serem enviados para uma página no servidor, pois, uma URL suporta no máximo 2.000 caracteres. Por exemplo, vamos supor que eu construísse uma página de um fórum para que o usuário escrevesse seu tópico, com certeza, jamais conseguiria fazer isso usando o método de envio GET, pois páginas de fórum deste tipo, possuem “normalmente” mais de 2.000 caracteres para serem enviados e processados.

Todas essas desvantagens do GET podem ser solucionadas através do uso do método de envio POST. Com o método POST, os dados enviados para uma página Web não são visualizados na URL do servidor, garantido assim que os dados não sejam capturados. E outra vantagem do POST é que você pode enviar dados “naturalmente” sem limite de tamanho .

1.5.2) Desenvolvendo a aplicação de envio de dados (usando método POST)

Vamos desenvolver agora a nossa aplicação baseada na técnica que envio de dados usando o método POST. Crie um projeto Android com os seguintes dados abaixo:

Application Name: EnviandoDadosPost

Project Name: EnviandoDadosPost



Package Name : com.example.enviandodadospost

Build SDK : Android 2.2 (API 8)

Minimum Required SDK : API 8: Android 2.2 (Froyo)

Activity Name: EnviandoDadosPostActivity

Layout Name : activity_enviando_dados_post

Title : Enviando Dados via POST

A estrutura da aplicação que vamos desenvolver é a mesma da aplicação anterior. Copie e cole o código abaixo no arquivo "activity_enviando_dados_post.xml":

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nome:"
    />

    <EditText
        android:id="@+id/ednome"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <requestFocus />
    </EditText>

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Idade:" />

    <EditText
```



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

```
        android:id="@+id/edidade"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/btenviar_dados"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Enviar dados" />

</LinearLayout>
```

Agora no arquivo “EnviandoDadosPostActivity.java”, vamos digitar o seguinte código abaixo:

```
package com.example.enviandodadospost;

import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
import android.widget.*;
import android.view.*;

import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;

import java.util.*;

public class EnviandoDadosPostActivity extends Activity {

    EditText ednome, edidade;

    Button btenviar_dados;

    int id;

    @Override
```



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.
        activity_enviando_dados_post);

    id = (int) Math.round(Math.random() * 10000) +
        10000;

    ednome = (EditText) findViewById(R.id.ednome);
    edidade = (EditText) findViewById(R.id.edidade);

    btenviar_dados = (Button) findViewById
        (R.id.btenviar_dados);

    btenviar_dados.setOnClickListener(new
        View.OnClickListener() {

        @Override
        public void onClick(View arg0) {

            try {

                String nome = ednome.getText().
                    toString();

                String idade = edidade.getText().
                    toString();

                HttpClient httpClient = new
                    DefaultHttpClient();

                HttpPost httpPost = new
                    HttpPost("http://www.livroaprendaandroid.
                        ueuo.com/cadastrar_dados_post.php");

                List<NameValuePair> nameValuePairs = new
                    ArrayList<NameValuePair>(3);

                nameValuePairs.add(new BasicNameValuePair
                    ("id",String.valueOf(id)));
                nameValuePairs.add(new BasicNameValuePair
                    ("nome",nome));
                nameValuePairs.add(new BasicNameValuePair
                    ("idade",idade));

                httpPost.setEntity(new
                    UrlEncodedFormEntity(nameValuePairs));

                httpClient.execute(httpPost);
```



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

```
ExibirMensagem("Dados enviados com  
sucesso. Seu ID para consulta é : " +  
id);  
  
        } catch (Exception e)  
        {  
            ExibirMensagem("Erro ao enviar  
dados");  
        }  
    }  
});  
}  
  
public void ExibirMensagem(String mensagem){  
  
    AlertDialog.Builder dialogo = new AlertDialog.  
Builder(EnviandoDadosPostActivity.this);  
    dialogo.setTitle("Aviso");  
    dialogo.setMessage(mensagem);  
    dialogo.setNeutralButton("OK", null);  
    dialogo.show();  
}  
  
}
```

Se observarmos o código da aplicação acima, é praticamente o mesmo da última aplicação, porém, com algumas diferenças (que envolve o uso da classe **HttpPost**). Vamos aos comentários:

A linha abaixo:

```
HttpClient httpClient = new DefaultHttpClient();
```

Cria uma instância da classe **HttpClient**, que funciona como se fosse uma “interface” cliente, que vai requisitar dados do servidor.



A linha abaixo:

```
HttpPost httpPost = new  
HttpPost("http://www.livroaprendaandroid.ueuo.com/cadastrar_dados_post  
.php");
```

Cria uma instância da classe **HttpPost**, responsável por enviar dados através do método POST. Observe que foi passado no parâmetro do seu construtor, o endereço da página que vai receber os dados através desse método.

A linha abaixo:

```
List<NameValuePair> nameValuePairs = new  
ArrayList<NameValuePair>(3);
```

Cria uma instância da classe **List**, que funciona como um “array” dinâmico que recebe valores. Os valores que serão armazenados nesse “array” são os valores que serão enviados pelo servidor, estruturados na classe **NameValuePair**. Observe que foi passado dois argumentos : o tipo de informação que será armazenado nesse Array (**NameValuePair**) e o valor 3, que corresponde aos três valores que serão enviados ao servidor, que são o “id”, “nome” e “idade”.

As instruções abaixo:

```
nameValuePairs.add(new BasicNameValuePair  
("id",String.valueOf(id)));  
  
nameValuePairs.add(new BasicNameValuePair("nome",nome));  
  
nameValuePairs.add(new BasicNameValuePair("idade",  
idade));
```

Adiciona no “array” os três valores que serão enviados para o servidor, que são o “id”, “nome” e “idade”, cada um com seus respectivos conteúdos.

A linha abaixo:

```
httpPost.setEntity(new UrlEncodedFormEntity  
(nameValuePairs));
```



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

Prepara os dados para serem enviados para o servidor. A classe **UrlEncodedFormEntity** é responsável por codificar a estrutura “nameValuePair” do tipo **List**, para que ela possa ser enviada para o servidor.

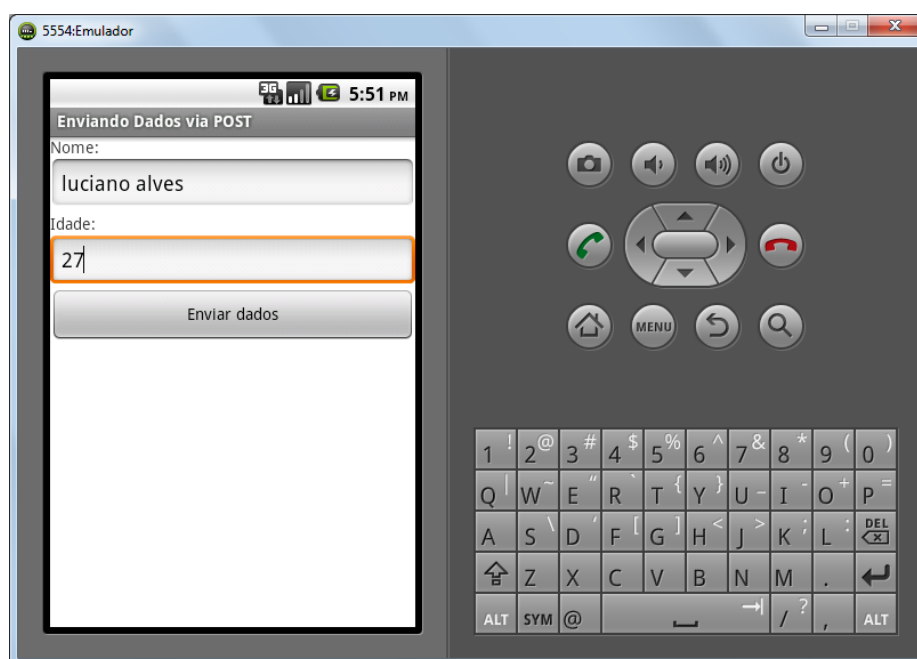
A linha abaixo:

```
httpClient.execute(httpPost);
```

Envia os dados para o servidor através do método **execute**, da classe **HttpClient**, passando como parâmetro o objeto da classe **HttpPost**.

Lembre-se de que você precisa configurar a “permissão” de acesso a Internet para esta aplicação.

Execute a aplicação. O resultado você confere na figura seguinte:



Aplicação em execução



1.6) Fazendo download de um arquivo

Nós aprendemos neste capítulo a desenvolver um programa que obtinha um conteúdo de um determinado arquivo (de conteúdo “texto”). Agora vamos aprender a obter um conteúdo (fazer um download) de um arquivo de formato qualquer (como uma imagem). Crie um novo projeto com os seguintes dados abaixo:

Application Name: FazendoDownload

Project Name: FazendoDownload

Package Name : com.example.fazendodownload

Build SDK : Android 2.2 (API 8)

Minimum Required SDK : API 8: Android 2.2 (Froyo)

Activity Name: DownloadActivity

Layout Name : activity_download

Title : Download de Arquivo

Após criar o projeto altere a estrutura da tela para **LinearLayout** (Vertical) e em seguida apague o componente **TextView** da tela. Feito isso adicione os seguinte componentes abaixo:

Button

Propriedade	Valor
Id	btdownload
Width	fill_parent
Text	Baixar imagem

TextView

Propriedade	Valor
Text	Imagem



ImageView

Propriedade	Valor
Id	imagem
Src	(Deixar sem imagem)
Height	wrap_content
Width	wrap_content

Depois de adicionar os componentes acima, vamos no arquivo “DownloadActivity.java”, para colocar o seguinte código abaixo:

```
package com.example.fazendodownload;

import java.io.InputStream;
import java.net.URL;

import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.widget.*;
import android.view.*;

public class DownloadActivity extends Activity {

    Button btbaixar_imagem;

    ImageView imagem;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_download);
        btbaixar_imagem = (Button) findViewById(
            R.id.btbaixar_imagem);
        imagem = (ImageView) findViewById(R.id.imagem);
        btbaixar_imagem.setOnClickListener(new View
            .OnClickListener() {

                @Override
                public void onClick(View arg0) {

                    try
                    {
```




Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

```
URL url = new URL("http://www.livroaprendaandroid.ueuo.com/icone-android.png");
InputStream is = url.openStream();

Bitmap bitmap = BitmapFactory.decodeStream(is);

imagem.setImageBitmap(bitmap);

} catch (Exception e) {

AlertDialog.Builder dialogo = new
AlertDialog.Builder(DownloadActivity.this);

dialogo.setTitle("Erro");
dialogo.setMessage("Erro ao fazer download da
imagem");
dialogo.setNeutralButton("OK", null);
dialogo.show();

}
}
});
}
}
```

Vamos analisar algumas linhas de código. A maioria dos códigos desse programa já são conhecidos. A linha abaixo:

```
Bitmap bitmap = BitmapFactory.decodeStream(is);
```

Cria um objeto do tipo **Bitmap** e carrega nele um objeto **InputStream**, que será convertido em um objeto de imagem, suportado pelo tipo do objeto.

Logo após o download da imagem, ela é visualizada através do código abaixo:

```
imagem.setImageBitmap(bitmap);
```

Só para lembrar que antes de executar a aplicação, você deve incluir a “permissão” de acesso a Internet. Feito isso a aplicação será executada com sucesso, conforme você pode conferir em seguida:



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)



Aplicação que faz downloads em execução

1.7) O componente WebView

Esse componente **WebView** é muito utilizado quando queremos desenvolver um navegador próprio para o nosso dispositivo. Embora o emulador já possua um Browser embutido, o Android já fornece um componente próprio voltado para tal finalidade. Ele funciona como um “display” onde podemos visualizar as páginas da Internet. Normalmente o componente **WebView** é trabalhado em conjunto com o componente **WebSettings**.

Vamos ver alguns métodos dos componentes:

Método	Descrição
<code>boolean canGoBack()</code>	Retorna verdadeiro caso seja possível voltar para a página anterior.
<code>boolean canGoForward()</code>	Retorna verdadeiro caso seja possível avançar para a próxima página.
	Retorna verdadeiro caso seja possível retornar para algumas páginas anteriores ou avançar alguns páginas. Ex: se você entrar com o valor 2, você



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

<code>boolean canGoBackOrForward(int passos)</code>	aqui verifica se é possível avançar duas páginas. Se você entrar com o valor -2, você aqui verifica se é possível retornar duas páginas anteriores. Logo, números positivos indicam números de páginas que você deseja avançar e números negativos indicam números de páginas que você deseja retornar.
<code>void clearHistory()</code>	Lista o histórico das páginas visitadas.
<code>WebSettings getSettings()</code>	Retorna um objeto WebSettings usado para fazer as configurações do WebView.
<code>String getTitle()</code>	Retorna o título da página corrente.
<code>String getUrl()</code>	Retorna a URL da página corrente.
<code>void goForward()</code>	Avança uma página.
<code>void goBack()</code>	Retorna à página anterior.
<code>void goBackOrForward(int passos)</code>	Retorna para algumas páginas anteriores ou avançar alguns páginas. Ex: se você entrar com o valor 2, você avança duas páginas. Se você entrar com o valor -2, você retorna duas páginas anteriores. Logo, números positivos indicam números de páginas que você deseja avançar páginas e números negativos retornam páginas anteriores.
<code>void loadUrl(String url)</code>	Carrega a url especificada no parâmetro.

- WebSettings

Métodos	Descrição
<code>void setSaveFormData (boolean arg)</code>	Define se o Browser deve salvar dados de formulário (true) ou não (false).
<code>void setJavaScriptEnabled(boolean</code>	Habilita (true) ou não (false) a execução de JavaScript existentes em algumas páginas



arg)	Web.
void setPassword(boolean arg)	Habilita (true) ou não (false) o armazenamento de senhas digitadas.

1.8) Desenvolvendo uma aplicação que visita páginas Web

Vamos criar agora um navegador utilizando o componente **WebView**. Crie um novo projeto com os seguintes dados abaixo:

Application Name: NavegadorWeb

Project Name: NavegadorWeb

Package Name : com.example.navegadorweb

Build SDK : Android 2.2 (API 8)

Minimum Required SDK : API 8: Android 2.2 (Froyo)

Activity Name: NavegadorActivity

Layout Name : activity_navegador

Title : Navegador Web

O único arquivo que iremos modificar neste projeto é o "NavegadorActivity.java". Nele iremos digitar o seguinte código:

```
package com.example.navegadorweb;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.*;

public class NavegadorActivity extends Activity {

    @Override
```



Apostila de Android

Programando Passo a Passo

Programação com Internet (Edição Completa)

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    WebView webView = new WebView(this);  
    WebSettings webSettings = webView.getSettings();  
    webSettings.setSavePassword(false);  
    webSettings.setSaveFormData(false);  
    webSettings.setJavaScriptEnabled(true);  
    webSettings.setSupportZoom(false);  
  
    webView.loadUrl("http://www.livroaprendaandroid.ueuo  
    .com");  
    setContentView(webView);  
}  
}
```

E para finalizar, não se esqueça de colocar a “permissão” de acesso a Internet. Vamos executar a nossa aplicação. O resultado você vê na figura abaixo:



Aplicação que navega na Internet em execução



Conclusão a respeito do material

Nesta apostila aprendemos a desenvolver aplicações em Android voltadas para trabalhar com Internet. No início deste material aprendemos o que vem a ser Sockets e que como esses recursos funcionam, também aprendemos a construir uma aplicação cliente em Android e um servidor em Java, em seguida aprendemos a desenvolver aplicações que enviam e recebem informações de uma página/serviço Web e também aprendemos, através de um componente do próprio Android, e desenvolvermos um navegador de Internet.

Espero que esse material lhe tenha sido útil.

Abraços