Arquivos Texto

Gravando e lendo dados.

Abordaremos um pouquinho de como podemos criar e carregar arquivos do tipo texto no Android armazenando estas informações no cartão SD do dispositivo. Montaremos um exemplo prático abordando recursos como: Criar e utilizar o evento Onclick() do botão, Utilização da classe "Java.IO" para manipulação de arquivos, entre outros recursos que poderão ser acompanhados ao decorrer da leitura.

Opções de Armazenamento

O Android nos fornece diversas formas para salvar os dados, sendo que a solução depende da necessidade, podendo assumir as seguintes formas:

SharedPreferences: Armazenar dados particulares primitivos em pares chave-valor.

Internal Storage: Armazenar dados privados na memória do dispositivo.

External Storage: Armazenar dados públicos sobre o armazenamento externo compartilhado.

SQLite Databases: Armazenar dados estruturados em um banco de dados.

Usaremos a opção "*External Storage*", que significa Armazenamento externo.

External Storage – Armazenamento Externo

Os dispositivos compatíveis com o sistema Android suportam uma memória externa compartilhada que podemos utilizar para diversas tarefas, como por exemplo manipular arquivos do tipo texto. Podendo ser um cartão SD ou uma memória interna não removível. Os arquivos salvos para o armazenamento externo são de leitura para todos podendo ser modificado pelo usuário.

Comandos úteis:

Environment.getExternalStorageState() : Comando necessário para verificar se a mídia está disponível.

Environment.getExternalStorageDirectory(): comando para abrir um diretório que representa a raiz do armazenamento externo, usando o diretório:

/Android/dados/<nome do pacote>/arquivos

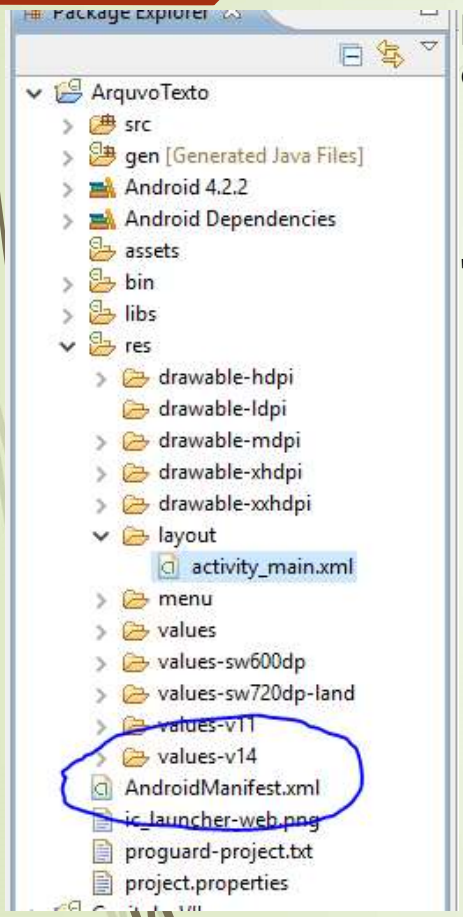
Criando um exemplo prático

O lay-out da aplicação se dividirá em duas partes, a primeira iremos criar um arquivo do tipo texto e salvá-lo no cartão SD, já a segunda escolheremos o arquivo para posteriormente carregá-lo em um EditText.
Para isto abra seu Eclipse e clique em "File/New/Android Application Project" de nome Arquivo Texto.

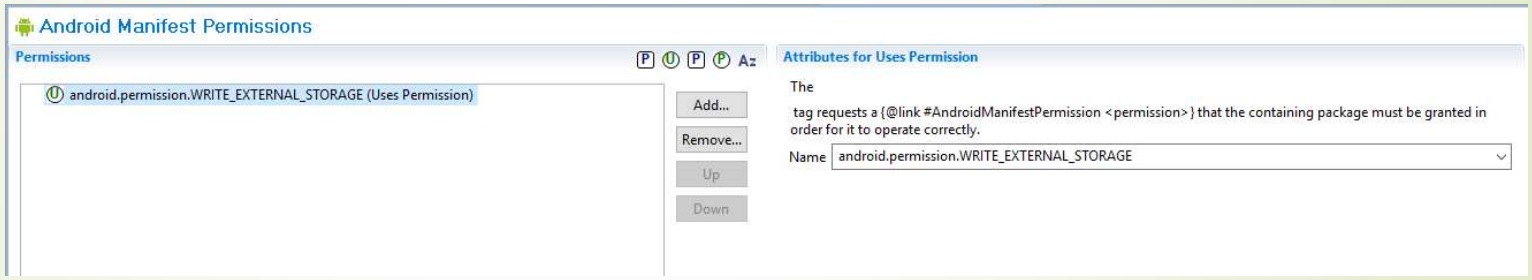
Crie uma imagem de um livro ou texto que desejar, com a extensão PNG , na terceira tela que carrega o ícone do aplicativo.

Adicionando permissões

Um detalhe importante para esse projeto é que será necessário habilitar a permissão "WRITE_EXTERNAL_STORAGE" no arquivo manifest, pois graças a essa permissão que seu aplicativo pode gravar arquivos no SD Card. Dê um duplo clique no arquivo "AndroidManifest.xml" e na aba "Permissions" clique no botão "Add" para adicionar o código a seguir. Ver Imagem 01.



"*android.permission.WRITE_EXTERNAL_STORAGE*"



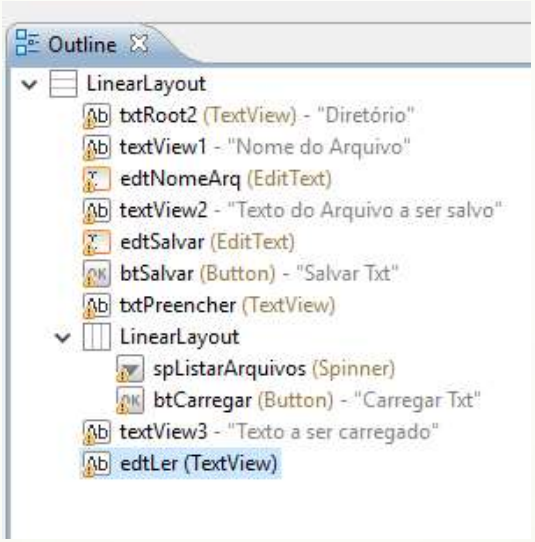
Esta permissão indica que podemos salvar dados em um dispositivo externo, o código deverá ficar parecido com o a seguir.

ArquvoTexto Manifest | activity_main.xml

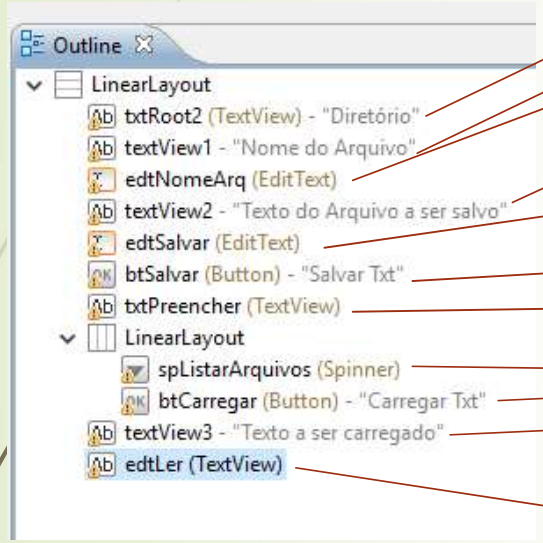
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.arquvotexto"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.arquvotexto.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

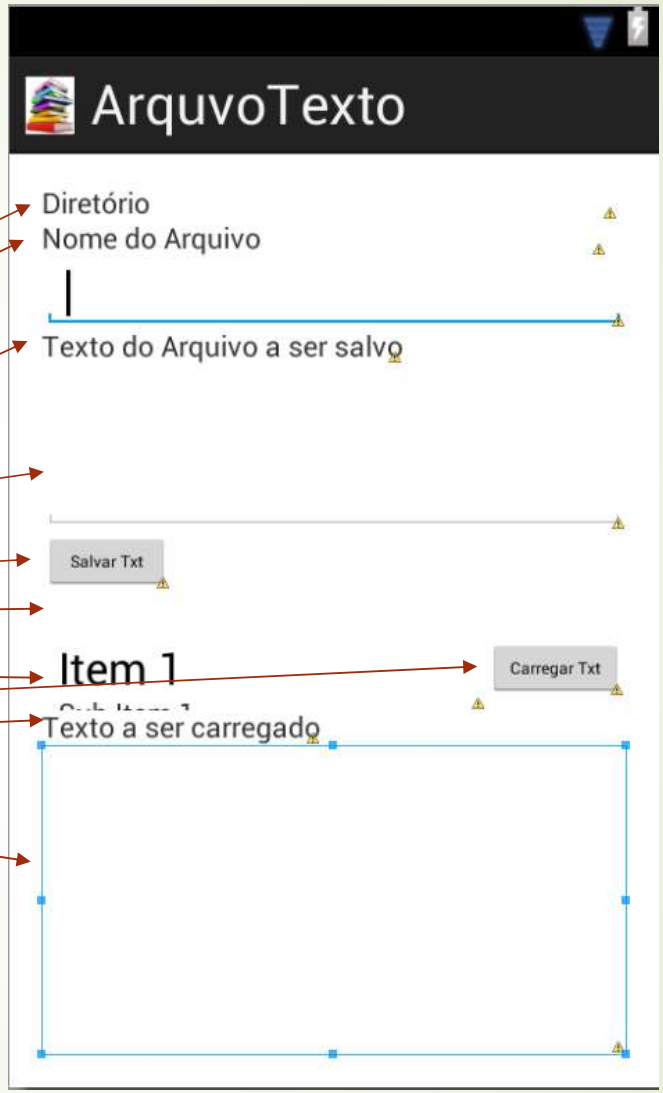
Criando a interface gráfica

Trabalharemos com os componentes padrões do Android, como: TextView, EditText, Button e Spinner. A imagem abaixo nos dá uma noção melhor de como os componentes deverão estar dispostos em nosso arquivo principal "main.xml".



Em tempo de programação, ficaremos com um layout semelhante ao que segue:





Codificando o Exemplo

Usaremos alguns pacotes adicionais, sendo necessário adicioná-los ao projeto.

```
import java.io.File;
import java.io.FileOutputStream;
import java.util.ArrayList;

import android.os.Bundle;
import android.os.Environment;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
```

Criaremos também algumas variáveis ao decorrer do desenvolvimento, precisaremos ler o front-end:


```
public class MainActivity extends Activity {
    private TextView txtRoot;
    private TextView txtNomeArq;
    private TextView txtSalvar;
    private TextView txtLer;
    private Spinner SpnListarArquivos;
    private ArrayList<String> Arquivos = new ArrayList<String>();
```

Achei necessário criar uma função Padrão Mensagem (), a qual será responsável notificar o usuário. Esta classe "Toast" se difere da "AlertDialog.Builder" pois a mesma apenas informa no rodapé inferior da tela do Android o ocorrido e logo em seguida desaparece, achei interessante usar este tipo de recurso para fins de aprendizado, podemos conferir a seguir o método que recebe como parâmetro uma String.

```
private void Mensagem(String msg)
{
    Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_SHORT).show();
}

private String ObterDiretorio()
{
    File root = android.os.Environment.getExternalStorageDirectory();
    return root.toString();
}
```

Este método nos retorna o diretório de armazenamento externo.

Criando o método Listar

O método Listar() preencherá o componente Spinner com os arquivos do tipo ".txt" salvos no diretório externo. Usamos os tipos de variáveis File e File[], sendo respectivamente responsáveis por obter o diretório e os arquivos deste diretório. Adicionamos os arquivos em um "Array" para posteriormente utilizá-los.

```
//###### MÉTODO LISTAR #########
public void Listar()
{
   File diretorio = new File(ObterDiretorio());
   File[] arquivos = diretorio.listFiles();
if(arquivos != null)
   {
      int length = arquivos.length;
      for(int i = 0; i < length; ++i)
      {
          File f = arquivos[i];
          if (f.isFile())
          {
              Arquivos.add(f.getName());
          }
      }

      ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>
                (this,android.R.layout.simple_dropdown_item_1line, Arquivos);
                    SpnListarArquivos.setAdapter(arrayAdapter);
   }
}
```

Criando o método Salvar()

Ao clicarmos no botão Salvar executaremos o método "Click_Salvar", usaremos um Try..Catch, onde transformamos o texto digitado em um "Array de Bytes" e com o método "Write" inserimos os dados seguido de uma notificação ao usuário. Com o método Listar() atualizaremos o Spinner com o nome dos arquivos ".txt".

```
//###### MÉTODO SALVAR #########
public void click_Salvar(View v)
{
    String lstrNomeArq;
    File arq;
    byte[] dados;
      try
      {
          lstrNomeArq = txtNomeArq.getText().toString();

          arq = new File(Environment.getExternalStorageDirectory(), lstrNomeArq);
                    FileOutputStream fos;

          dados = txtSalvar.getText().toString().getBytes();

          fos = new FileOutputStream(arq);
          fos.write(dados);
          fos.flush();
          fos.close();
          Mensagem("Texto Salvo com sucesso!");
          Listar();
      }
      catch (Exception e)
      {
        Mensagem("Erro : " + e.getMessage());
      }
}
```

No evento OnCreate() faremos atribuições às variáveis e invocaremos o método Listar() seguido de um Try..Catch


```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    try {

        txtRoot = (TextView) findViewById(R.id.txtRoot2);
        txtNomeArq = (TextView) findViewById(R.id.edtNomeArq);
        txtSalvar = (TextView) findViewById(R.id.edtSalvar);
        txtLer = (TextView) findViewById(R.id.edtLer);
        SpnListarArquivos = (Spinner)  findViewById(R.id.spListarArquivos);
        txtRoot.append(ObterDiretorio());

        Listar();

    } catch (Exception e) {
       Mensagem("Erro : " + e.getMessage());
    }
}
```

Criando o método Carregar()

Já o método carregar, continuaremos a utilizar o Try..Catch, sendo que de primeiro momento pegamos o item que está selecionado no componente Spinner, limpamos o campo "txtLer" e logo em seguida efetuamos uma leitura linha a linha do arquivo carregando para a caixa de texto toda a informação lida.

```
//######## MÉTODO CARREGAR TEXTO SELECIONADO ########
public void click_Carregar(View v)
{
    String lstrNomeArq;
    File arq;
    String lstrlinha;
     try
     {
         lstrNomeArq = SpnListarArquivos.getSelectedItem().toString();
         txtLer.setText("");

         arq = new File(Environment.getExternalStorageDirectory(), lstrNomeArq);
         BufferedReader br = new BufferedReader(new FileReader(arq));

          while ((lstrlinha = br.readLine()) != null)
          {
               if (!txtLer.getText().toString().equals(""))
                {
                    txtLer.append("\n");
                }
                txtLer.append(lstrlinha);
          }

          Mensagem("Texto Carregado com sucesso!");

      }
      catch (Exception e)
      {
         Mensagem("Erro : " + e.getMessage());
      }
}
```