


✓ Introdução à Programação Android

Tutorial: Persistência de dados

Objetivo: Salvar informações de forma persistente no Android, utilizando Shared Preferences ou o Storage (interno ou externo) do dispositivo.

- 1) Abra no Eclipse o projeto criado no tutorial de troca e telas.
- 2) Modifique a tela principal de forma semelhante conforme apresentado abaixo:

3G 4:56

 persist

Texto

Números

Salvar onde?

☐ Shared Preferences

☐ Internal Storage

Salvar os dados

Exibir

- 3) Defina um método para o botão “Salvar os dados”: Clique com o botão direito do mouse em cima do botão e selecione: opção “Other Properties” → “Inherited from View” → “OnClick”.

DICA: fique atento aos Ids de cada componente para que eles sejam referenciados corretamente no código.

```
public void botaoSalvarDados(View view) {
```

```

        // Instância do elemento de opção.
        RadioButton opcaoSharedPreferences = (RadioButton)
findViewById(R.id.radioButtonShared);

        // Recuperamos os valores dos campos da tela.
        EditText campoTexto = (EditText) findViewById(R.id.editText1);
        EditText campoNumero = (EditText)
findViewById(R.id.editText2);

        String texto = campoTexto.getText().toString();
        long numero =
Long.parseLong(campoNumero.getText().toString());

        // Teste da opção selecionada.
        if (opcaoSharedPreferences.isChecked()) {
            salvarShared(texto, numero);
        } else {
            salvarInternalStorage(texto, numero);
        }
    }
}

```

Salvando com Shared Preferences

Utilizando o recurso Shared Preferences, é possível salvar entradas do tipo chave-valor, onde se associa um “nome” a uma determinada informação para que depois se possa recuperá-la justamente através deste nome. O Android salva tudo em um arquivo XML dentro da estrutura interna “de disco” em cada aplicação.

4) Implemente a função abaixo para fazer a escrita das informações:

- ✓ DICA: Opção indicada para poucas informações e que sejam simples, como números, strings e valores booleanos.

```

private void salvarShared(String texto, long numero) {

    // Cria ou abre.
    SharedPreferences prefs =
getSharedPreferences("preferencias_1", Context.MODE_PRIVATE);

    // Precisamos utilizar um editor para alterar Shared
Preferences.
    Editor ed = prefs.edit();

    // salvando informações de acordo com o tipo
    ed.putString("TEXTO", texto);
    ed.putLong("NUMERO", numero);

    // Grava efetivamente as alterações.
    ed.commit();
}

```

Salvando com Storage

A opção do Storage é um espaço “em disco” que cada aplicação tem onde é possível salvar arquivos. Existe a opção do Internal Storage (espaço na estrutura de arquivos interna da

aplicação) e do External Storage, que geralmente é um espaço no SD Card – podendo ser público (pastas de música ou fotos por exemplo) ou da aplicação.

5) Implemente a função abaixo para salvar as informações na memória interna do dispositivo.

```
private void salvarInternalStorage(String texto, long numero) {
    FileWriter fileWriter = null;

    try {
        // Cria o arquivo onde serão salvas as informações.
        File file = new
File(getFilesDir().getPath()+"/arquivo_1.txt");
        fileWriter = new FileWriter(file, true);

        fileWriter.append(texto);
        fileWriter.append("\n");// Quebra de linha.
        fileWriter.append(String.valueOf(numero));

        // Escreve no arquivo.
        fileWriter.flush();

    } catch (IOException e) {
        Log.e("Erros", "Erro ao salvar usando Internal
Storage", e);
    } finally {
        // Fecha os recursos.
        if (fileWriter != null) {
            try {fileWriter.close();}
            catch(Exception e){}
        }
    }
}
```

6) Implemente também a função mostrada abaixo para salvar os dados na memória externa do dispositivo (SD Card):

```
private void salvarExternalStorage(String texto, long numero) {

    // Obtém o estado do storage.
    String mediaState = Environment.getExternalStorageState();

    // Testa se ele está disponível.
    if (mediaState.equals(Environment.MEDIA_MOUNTED)) {

        FileWriter fileWriter = null;
        try {
            // Cria o arquivo onde serão salvas as
informações.
            File file = new
File(getExternalFilesDir(null).getPath(), "/arquivo_2.txt");
            fileWriter = new FileWriter(file, true);

            fileWriter.append(texto);
            fileWriter.append("\n");// Quebra de linha.
            fileWriter.append(String.valueOf(numero));

            // Escreve no arquivo.
            fileWriter.flush();

        } catch (IOException e) {
```

```

        Log.e("Erros", "Erro ao salvar usando External
Storage", e);
    } finally {
        // Fecha os recursos.
        if (fileWriter != null) {
            try {fileWriter.close();}
            catch(Exception e){}
        }
    }
}
}
}

```

7) Adicione a seguinte permissão ao arquivo AndroidManifest.xml , para que o aplicativo possa ter acesso ao cartão de memória.

```

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-
permission>

```

8) Defina também um método para o botão “Exibir”: Clique com o botão direito do mouse em cima do botão e selecione: opção “Other Properties” → “Inherited from View” → “OnClick”. Implemente a função abaixo dentro do arquivo MainActivity.java

```

public void botaoExibirDados(View view) {
    // Instância do elemento de opção
    RadioButton opcaoSharedPreferences = (RadioButton)
findViewById(R.id.radioButtonShared);

    // Cria o intent indicando qual a opção foi usada para salvar
os dados.
    Intent intent = new Intent(this, SegundaActivity.class);

    if (opcaoSharedPreferences.isChecked())
        intent.putExtra("opcao", "shared_pref");
    else
        intent.putExtra("opcao", "int_storage");

    // Inicia a nova tela.
    startActivity(intent);
}

```

9) Para exibir os dados na segunda tela, implemente o seguinte código dentro da classe SegundaActivity.java:

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Recupera o intent que iniciou a atividade e a mensagem.
    Intent intent = getIntent();
    String opcao = intent.getStringExtra("opcao");

    String conteudoSalvo;
    if (opcao.equalsIgnoreCase("shared_pref")) {
        conteudoSalvo = acessaSharedPreferences();
    } else {
        conteudoSalvo = acessaInternalStorage();
    }
}

```

```

// Cria uma view para exibir as informações salvas.
TextView textView = new TextView(this);
textView.setTextSize(30);
textView.setText(conteudoSalvo);

// Define que o conteúdo exibido pela tela é o campo que
// irá exibir as informações.
setContentView(textView);
}

```

10) Para lermos os dados salvos utilizando o Shared Preferences, implemente a seguinte função:

```

private String acessaSharedPreferences() {
    // Acesso às Shared Preferences usando o nome definido.
    SharedPreferences prefs = getSharedPreferences("preferencias_1",
Context.MODE_PRIVATE);

    // Acesso às informações de acordo com o tipo.
    String texto = prefs.getString("TEXTO", "não encontrado");
    long numero = prefs.getLong("NUMERO", 0);

    // Formata um string com todo o conteúdo separado por linha.
    return (texto + "\n" + numero);
}

```

11) A outra opção é lermos os dados salvos utilizando o Internal Storage. Para isso, implemente a seguinte função:

```

private String acessaInternalStorage() {

    String line;
    String conteudo = "";

    BufferedReader br = null;
    try {
        // Acessa o arquivo.
        br = new BufferedReader(new
FileReader(getFilesDir().getPath()+"/arquivo_1.txt"));

        // Faz a leitura, uma linha por vez, até o fim do arquivo,
        // gerando um string com todo o conteúdo separado por linha.
        while ((line = br.readLine()) != null) {
            conteudo += line;
            conteudo += "\n"; // Quebra de linha.
        }

    } catch (Exception e) {
        Log.e("Erros", "Erro ao ler arquivo do Internal Storage", e);
    } finally {
        if (br != null) {
            try {br.close();}
            catch (Exception e){}
        }
    }

    // Retorna o conteúdo do arquivo.
    return conteudo;
}

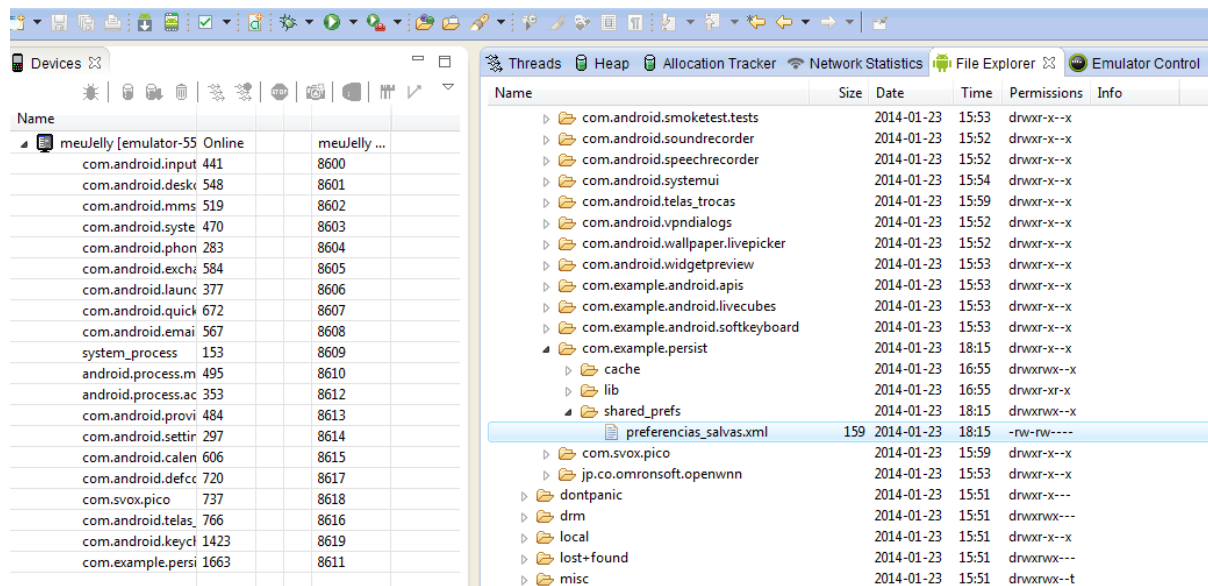
```

}

12) Inicie um emulador. Após este ter iniciado, clique com o botão direito na pasta raiz do projeto e escolha “Run as” – “Android Application”.

13) Teste seu aplicativo salvando alguns dados.

DICA: No emulador ou em um dispositivo com “root” é possível ver o sistema de arquivos da aplicação no Eclipse, através da perspectiva DDMS, na aba File Explorer. Cada aplicativo possui uma pasta com o seu nome no caminho /data/data/.



TRABALHO

Faça um aplicativo que exiba campos para o preenchimento de informações de bandas de música: Gênero Musical, Grupo, Álbum, Faixas. O software também deve salvar estas informações em um arquivo.txt na memória externa (SD Card) do dispositivo.

DICA 1: Para testar o uso do External Storage, apenas substitua os métodos de escrita e leitura no código da aplicação criado no tutorial.

DICA 2: A leitura do External Storage é muito parecida com a que utilizamos no Internal, bastando testar a disponibilidade e usar o caminho correspondente. A pasta onde os arquivos são salvos nesse caso é a /mnt/sdcard/Android/data//files/.