Notepad – Exercício 02

Nesse exercício vamos adicionar uma segunda atividade à aplicação de notepad desenvolvida no exercício anterior. Recomendo que o faça para poder entender o que vai acontecer aqui. Essa nova atividade permitirá ao usuário criar e editar notas. Também permitirá ao usuário excluir notas por meio de um menu de contexto. A nova atividade assumirá a responsabilidade de criar novas notas coletando os dados de entrada e empacotando-os em um Bundle que será retornado por um intent.

O exercício demonstrará:

- Como construir uma nova atividade e adicioná-la ao manifesto do Android;
- Fazer a chamada de outra atividade de modo assíncrono usando startActivityForResult();
- Passar dados entre atividades dentro de objetos Bundle;
- Como fazer uso de layouts de tela mais avançados e;
- Como criar um menu de contexto.

Passo 1

Abra o projeto Notepadv2 que você baixou da pasta Arquivos e você poderá perceber o seguinte:

Quando abrir o arquivo string.xml dentro de res/values perceberá inúmeros novos valores string que serão usados em novas funcionalidades.

Também, abrindo a classe Notepadv2, você notará muitas novas constantes definidas com o novo campo mNotesCursor usado para guardar o cursor que estaremos usando. Note também que o método fillData() tem mais comentários e agora usa novos campos para guardar informações do Cursor de notas. O método onCreate() está do mesmo jeito que no primeiro exercício. Também note que o campo membro usado para guardar o Cursor de notas agora é chamado de mNotesCursor. O m à frente do campo denota que ele é um campo membro e é parte do padrão de codificação do Android utilizado ainda atualmente.

Existem ainda outros métodos que foram sobrescritos (onCreateContextMenu(), onContextItemSelected(), onListItemClick() e onActivityResult()).

Passo 2

Primeiro, vamos criar o menu de contexto que permitirá aos usuários excluir notas individuais. Abra a classe Notepadv2.

Para que cada item da lista na ListView se registre para o menu de contexto, nós chamamos registerForContextMenu() e passamos para a ListView. Então, ao final do método onCreate(), adicionamos essa linha:

```
registerForContextMenu(getListView());
```

Como nossa atividade extends a classe de ListActivity, getListView(), retornará o objeto ListView local para a atividade. Agora, cada item na ListView vai ativar o menu de contexto. Vamos preencher o método onCreateContextMenu(). Aqui, adicionamos apenas uma linha, que adicionará um novo item de menu para excluir a nota. Chame menu.add() como mostrado abaixo:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenuInfo menuInfo) {
   super.onCreateContextMenu(menu, v, menuInfo);
   menu.add(0, DELETE_ID, 0, R.string.menu_delete);
}
```

O callback onCreateContextMenu() passa outras informações para o objeto de menu, como a View que está chamando o menu e um objeto extra que pode conter informações adicionais sobre o objeto selecionado. Contudo, nós não nos importamos com isso agora, já que apenas um tipo de objeto na atividade usa o menu de contexto.

Passo 3

Agora que você já registrou nosso ListView para o menu de contexto e definiu nossos itens de menu de contexto, precisamos processar o callback quando selecionado. Para isso, precisamos identificar o ID de lista do item selecionado e então excluí-lo. Então, preencha o método onContextItemSelected() como mostrado abaixo:

```
@Override
public boolean onContextItemSelected(MenuItem item) {
   switch (item.getItemId()) {
       case DELETE_ID:
                       AdapterView.AdapterContextMenuInfo info =
(AdapterView.AdapterContextMenuInfo) item.getMenuInfo();
           mDbHelper.deleteNote(info.id);
           fillData();
           return true;
   }
   return super.onContextItemSelected(item);

}
```

Aqui, nós retornamos o AdapterContextMenuInfo com o getMenuInfo(). O campo id do objeto mostra a posição do item no ListView. Nós então passamos ele ao método deleteNote() de nosso NotesDbAdapter e a nota será excluída. Notas agora podem ser excluídas.

Passo 4

Preencha o corpo do método createNote().

Crie um novo Intent para criar uma nota (ACTIVITY_CREATE) usando a classe NoteEdit. Então chame o Intent usando o método startActivityForResult():

```
Intent i = new Intent(this, NoteEdit.class);
startActivityForResult(i, ACTIVITY_CREATE);
```

Essa forma de chamada do Intent tem como alvo uma classe específica de nossa atividade, a NoteEdit. Já que a classe Intent precisará comunicar-se com o sistema Android para rotear requisições, nós também temos de prover um Context(this)

O método startActivityForResult() chama o Intent de uma maneira que causa a

chamada de um método em nossa atividade quando a nova atividade é completada. O método em nossa atividade que recebe o callback é chamado de onActivityResult() e nós vamos implementá-lo em um passo posterior. A outra maneira de chamar uma atividade é usando startActivity() mas essa é uma chamada do tipo 'chame-e-esqueça' - dessa maneira nossa atividade não é informada quando a atividade foi completada e não existe nenhuma maneira de retornar a informação de resultado da atividade chamada com o startActivity().

Não se preocupe que a atividade NoteEdit não exista ainda. Mas à frente vamos criá-la.

Passo 5

Preencha o corpo do método onListItemClick().

Esse método é chamado quando o usuário seleciona um item da lista. É passado a ele quatro parâmetros: o objeto ListView que foi chamado, a View dentro da ListView que foi clicada, a posição na lista na qual foi clicada e a mRowId do item que foi clicado. Nessa instância podemos ignorar os dois primeiros parâmetros (nós temos apenas um único ListView que pode ser usado) e nós ignoramos a mRowId também. Tudo que nos interessa é a posição que o usuário selecionou. Nós estamos usando esse dado para buscar os dados da linha correta e inseri-lo num Bundle para ser enviado à atividade NoteEdit.

Em nossa implementação do callback, o método cria um Intent para editar a nota usando a classe NoteEdit. Ele então adiciona dados em Bundles extras do Intent que vamos passar para a atividade chamada. Nós o usamos para passar o título e texto do corpo e a mRowId para a nota que nós estamos editando. Finalmente, chamamos o intent usando startActivityForResult(). Aqui está o código que pertence ao onListItemClick():

```
super.onListItemClick(l, v, position, id);
Cursor c = mNotesCursor;

c.moveToPosition(position);
Intent i = new Intent(this, NoteEdit.class);

i.putExtra(NotesDbAdapter.KEY_ROWID, id);
i.putExtra(NotesDbAdapter.KEY_TITLE, c.getString(
        c.getColumnIndexOrThrow(NotesDbAdapter.KEY_TITLE)));
i.putExtra(NotesDbAdapter.KEY_BODY, c.getString(
        c.getColumnIndexOrThrow(NotesDbAdapter.KEY_BODY)));
startActivityForResult(i, ACTIVITY_EDIT);
```

putExtra() é o método que adiciona itens aos Bundles extras para passá-los em chamadas de Intent. Aqui nós estamos usando um Bundle para passar o título, corpo e mRowId da nota que nós podemos editar. Os detalhes da nota são capturados a partir de uma pesquisa no Cursor que movemos para uma posição apropriada para o elemento selecionado na lista com o método moveToPosition().

Com os extras adicionados ao Intent, nós chamamos o Intent na classe NoteEdit passando ao startActivityForResult() o Intent e o código de requisição (esse código será retornado para o onActivityResult como parâmetro requestCode).

Nota: Nós assinalamos o mNotesCursor para uma variável local no início do método. Isso é feito assim como uma otimização do código Android. Acessar variáveis locais é muito mais eficiente do que acessar um campo na máquina virtual Dalvik. Então, fazendo isso, temos apenas um acesso ao campo na máquina virtual Dalvik e cinco acessos à variável local, fazendo da rotina algo muito mais eficiente. É recomendado que você use esse tipo de otimização quando possível.

Passo 6

Os métodos createNote() e onListItemClick() usam uma chamada de Intent assíncrona. Nós precisamos de um handler para o callback. Por isso, aqui preenchemos o corpo de onActivityResult().

onActivityResult() é um método sobrescrito que será chamado quando a atividade retornar com um resultado (lembre-se, uma atividade vai apenas retornar se lançada com startActivityForResult()). Os parâmetros providos ao callback são:

- **requestCode –** o código original de requisição para uma invocação de Intent (ou ACTIVITY_CREATE ou ACTIVITY_EDIT).
- **resultCode –** o resultado (ou código de erro) da chamada. Deverá retornar 0 se tudo correu bem ou deverá haver um valor diferente de 0 indicando que algum erro aconteceu. Existem códigos de erro padrão e você poderá criar suas próprias constantes para indicar erros específicos.
- **intent –** esse é o Intent criado pela atividade que retorna os resultados. Ele pode ser usado para retornar dados em intents “extras”.

A combinação de startActivityForResult() e onActivityResult() pode ser pensada como uma chamada assíncrona RPC (remote procedure call) e forma a maneira recomendada para uma atividade chamar outra atividade e compartilhar serviços.

Aqui está o código que deverá estar em onActivityResult():

```
super.onActivityResult(requestCode, resultCode, intent);
Bundle extras = intent.getExtras();

switch (requestCode) {
   case ACTIVITY_CREATE:
                                          String      title      =
extras.getString(NotesDbAdapter.KEY_TITLE);
       String body = extras.getString(NotesDbAdapter.KEY_BODY);
       mDbHelper.createNote(title, body);
       fillData();
       break;
   case ACTIVITY_EDIT:
       Long mRowId = extras.getLong(NotesDbAdapter.KEY_ROWID);
       if (mRowId != null) {
                                          String    editTitle    =
extras.getString(NotesDbAdapter.KEY_TITLE);
                                           String    editBody    =
extras.getString(NotesDbAdapter.KEY_BODY);
           mDbHelper.updateNote(mRowId, editTitle, editBody);
       }
       fillData();
       break;
}
```

Estamos gerenciando ambos resultados de atividade com esse método, tanto o do ACTIVITY_CREATE quanto do ACTIVITY_EDIT.

No caso de ser create, nós capturamos o título e o corpo a partir dos extras (retornados de um Intent que foi retornado) e os usamos para criar uma nota.

No caso de ser edit, nós capturamos o mRowId também e o usamos para fazer o update da nota no banco de dados.

fillData() no final certifica que todo o conteúdo mostrado está atualizado.

Passo 7

Abra o arquivo note_edit.xml que está no projeto e dê uma olhada nele. Esse será o código fonte para o Note Editor.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/title" />
        <EditText
            android:id="@+id/title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1" />
    </LinearLayout>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/body" />
    <EditText
        android:id="@+id/body"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:scrollbars="vertical" />
    <Button
        android:id="@+id/confirm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/confirm" />
</LinearLayout>
```

Existe um novo parâmetro usado aqui que nós ainda não vimos anteriormente: android:layout_weight. Ele está setado para o valor 1 caso em que o usamos acima.

O layout_weight é usado em layouts linears (LinearLayout) para assinalar a importância da view dentro do layout.

Passo 8

Crie a classe NoteEdit que extends AppCompatActivity.

Utilize o “wizard” e desmarque a opção “generate a Layout File” e pressione “finish”. Agora sobrescreva o método onCreate()

Passo 9

Preencha o corpo do método onCreate() para o NoteEdit.

Aqui é onde vamos setar o título de nossa nova atividade para, digamos, “Editar Nota” (ou qualquer valor que exista em strings.xml para essa opção). Vamos também setar o content view para usar nosso arquivo de layout note_edit.xml. Nós também buscaremos handles para o título e corpo e o botão de confirmação para que nossa classe possa usá-los para capturar e modificar títulos de nota e corpo e finalmente anexá-los ao evento de confirmação quando o usuário pressionar o botão correto.

Então nós vamos separar os valores que foram passados para a Atividade com os Bundles extras anexados ao intent chamado. Vamos usá-los para pre-popular o título e texto do corpo da nota para que os usuários possam editá-los. Então vamos buscar e guardar o mRowId para que possamos saber que nota o usuário está editando.
Dentro de onCreate(), vamos dizer qual layout usar:

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

Mudamos o título da atividade para o valor da string edit_note que se encontra em strings.xml

```
setTitle(R.string.edit_note);
```

Encontramos os componentes EditText e Button que precisamos:

```
mTitleText = findViewById(R.id.title);
mBodyText = findViewById(R.id.body);
Button confirmButton = findViewById(R.id.confirm);
```

Note que mTitleText e mBodyText são campos membros (eles devem ser declarados no topo da definição de classe).

No topo da classe, declare um campo Long mRowId privado para guardar o mRowId atual que está sendo editado.

Continuando dentro de onCreate(), adicione o código para inicializar o title, body e mRowId a partir dos Bundles extras do Intent (se ele estiver presente):

```
mRowId = null;
Bundle extras = getIntent().getExtras();
if (extras != null) {
   String title = extras.getString(NotesDbAdapter.KEY_TITLE);
   String body = extras.getString(NotesDbAdapter.KEY_BODY);
   mRowId = extras.getLong(NotesDbAdapter.KEY_ROWID);

   if (title != null) {
       mTitleText.setText(title);
   }
   if (body != null) {
       mBodyText.setText(body);
   }
}
```

Nós estamos capturando title e body a partir dos Bundles extras que foram enviados pela chamada do Intent. Estamos também protegendo os campos textos de receber valores nulos.

Criar um onClickListener para o botão:

Listeners podem ser um dos mais confusos aspectos da implementação da interface, mas nós estamos tentando chegar a esse objetivo de maneira simples. Queremos que o método onClick seja passado quando o usuário pressionar o botão de confirmação e usá-lo para realizar algum trabalho e retornar os valores da nota editada para quem chama o Intent. Fazemos isso usando uma coisa chamada anonymous inner class. É um pouco confuso de se ver a não ser que já tenhamos tido contado antes, mas se você não quer usar isso, você terá de verificar no futuro um código Java para como criar um Listener e anexá-lo a um botão. Aqui vai o listener vazio:

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

```
    public void onClick(View view) {
    }
});
```

Passo 10

Preencha o corpo do método onClick() do OnClickListener criado no último passo.

Esse é o código que deverá rodar quando o usuário clica no botão de confirmação. Nós queremos que ele pegue o título e texto do corpo a partir dos campos da tela e colocá-los em um Bundle para retorná-lo para a atividade que invoca o NoteEdit. Se a operação é uma edição em vez de criação, nós também vamos querer colocar o mRowId no Bundle para que a classe Notepadv2 possa salvar as mudanças de volta na nota correta.

1. Crie um Bundle e coloque o título e texto do corpo usando as constantes definidas em Notepadv2 como chaves:

```
Bundle bundle = new Bundle();

bundle.putString(NotesDbAdapter.KEY_TITLE,
mTitleText.getText().toString());
bundle.putString(NotesDbAdapter.KEY_BODY,
mBodyText.getText().toString());
if (mRowId != null) {
    bundle.putLong(NotesDbAdapter.KEY_ROWID, mRowId);
}
```

2. Configure a informação de resultado (o Bundle em si) como um novo Intent e finalize a atividade.

```
Intent mIntent = new Intent();
mIntent.putExtras(bundle);
setResult(RESULT_OK, mIntent);
finish();
```

O intent é simplesmente um compartimento carregando nossos Bundles (com o título, corpo e mRowId).

O método setResult() é usado para setar o código de resultado e retornar o Intent para ser passado de volta para o Intent que fez a chamada. No caso de tudo correr

bem, retornamos RESULT_OK como código de resultado.

A chamada finish() é usada para assinalar que a Atividade está pronta (como uma chamada de retorno). Qualquer coisa no resultado será retornado a quem chamou o Intent junto ao controle da execução.

O código completo de onCreate() (junto com os campos de classe de suporte a ela) deve se parecer como abaixo:

```
private Long mRowId;
private EditText mTitleText;
private EditText mBodyText;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.note_edit);
    setTitle(R.string.edit_note);

    mTitleText = findViewById(R.id.title);
    mBodyText = findViewById(R.id.body);
    Button confirmButton = findViewById(R.id.confirm);

    mRowId = null;
    Bundle extras = getIntent().getExtras();
    if (extras != null) {
                                            String      title      =
extras.getString(NotesDbAdapter.KEY_TITLE);
        String body = extras.getString(NotesDbAdapter.KEY_BODY);
        mRowId = extras.getLong(NotesDbAdapter.KEY_ROWID);

        if (title != null) {
            mTitleText.setText(title);
        }
        if (body != null) {
            mBodyText.setText(body);
        }
    }

    confirmButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
            Bundle bundle = new Bundle();

                        bundle.putString(NotesDbAdapter.KEY_TITLE,
mTitleText.getText().toString());
                         bundle.putString(NotesDbAdapter.KEY_BODY,
mBodyText.getText().toString());
```

```
            if (mRowId != null) {
                bundle.putLong(NotesDbAdapter.KEY_ROWID, mRowId);
            }

            Intent mIntent = new Intent();
            mIntent.putExtras(bundle);
            setResult(RESULT_OK, mIntent);
            finish();
        }
    });

}
```

Passo 11

Finalmente, uma nova atividade tem de ser definida no arquivo de manifesto.

Antes da nova atividade ser vista pelo Android, ela precisa ter sua entrada própria no arquivo AndroidManifest.xml. Isso é para fazer com que o sistema saiba especificar quais IntentFilters a atividade implementa aqui, mas nós vamos deixar isso por agora e apenas fazer com que o Android saiba que a atividade foi definida.

Abra o arquivo AndroidManifest.xml clicando duas vezes sobre a pasta.

Verifique se a activity foi declarada no Manifest e caso não tenha sido criada automaticamente, faça a declaração dela no arquivo e feche para salvar as alterações.

Passo 12

Agora, é só rodar e correr para o abraço. Se tudo correr bem, uma tela como a que será mostrada pelo professor deverá aparecer no device.