03

Passando dados entre Activities

Transcrição

Conseguimos estabelecer uma interação quando o aluno é clicado. O que queremos fazer depois de clicarmos no nome do aluno é que a aplicação nos leve de volta ao formulário para que possamos editá-lo. Para fazer isso vamos acessar o comportamento da classe, no *ListaAlunosActivity.java*.

Lembrando que inserimos um `setOnClickListener` e imprimimos através do *Toast* uma mensagem de que o aluno foi clicado. Vamos apagar a linha do *Toast*, pois ela não será mais necessária:

```
Toast.makeText(ListaAlunosActivity.this, "Aluno " + aluno.getNome() + " clicado!", Toast.LENGHT_SHORT).show()
```

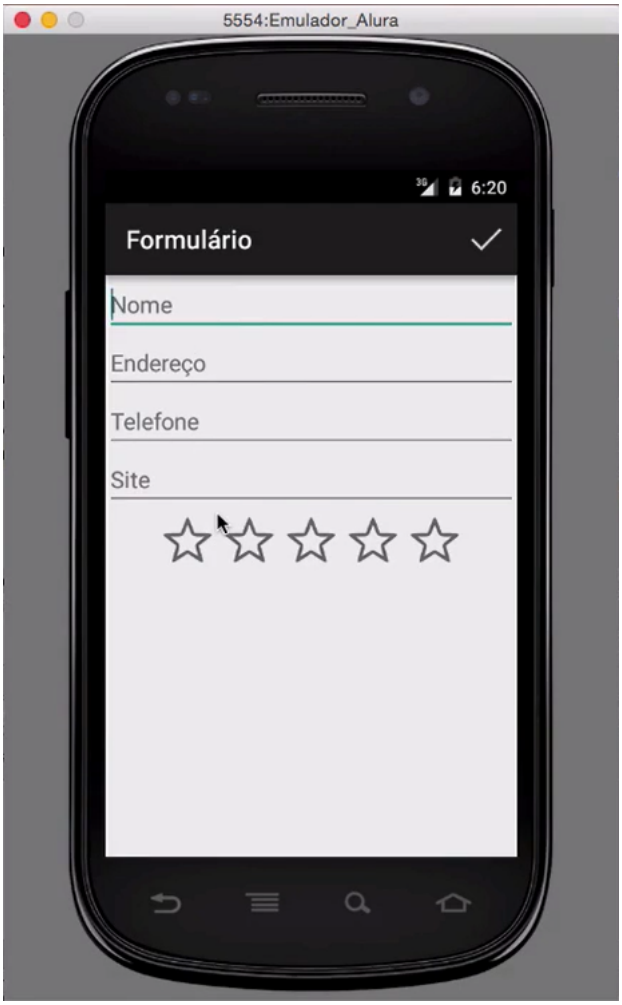
O que queremos é, ao clicar no aluno, ir para a tela do formulário. Para isso usaremos uma `intent` na qual declararemos ao *Android* que queremos mudar de tela. Escreveremos a `intent` na última linha da classe `setOnItemClickListener`. Lembrando que a `intent` pede duas informações: um contexto, que será `ListaAlunosActivity.this`, e a classe do formulário, `FormularioActivity.class`. Ficaremos com, `Intent intentVAiProFormulario = new Intent(ListaAlunosActivity.this, FormularioActivity.class)`. Para abrir a tela completaremos com um `startActivity`. Teremos `startActivity(intentVaiProFormulario)`. Nosso código estará da seguinte maneira:

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

        listaAlunos = (ListView) findViewById(R.id.lista_alunos);

        listaAlunos.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemCIick(AdapterView<?> lista, View item, int position, long id) {
                Aluno aluno = (Aluno) listaAlunos.getItemAtPosition(position);
                Intent intentVaiProFormulario = new Intent(ListaAlunosActivity.this, FormularioA
            }
        });
...//
    }
```

Vamos ver o que acontece no emulador?



O formulário abriu, mas falta recuperar os dados do aluno para que apareçam no formulário.

Vamos voltar na *ListaAlunosActivity.java* e como já recuperamos o aluno, `Aluno aluno` , em um momento anterior, vamos transmiti-lo para o *FormularioActivity.java*.

A `intent` é usada para demostrar intenção. Ao criarmos a `intent` , o *Android* interpreta que a *ListaAlunosActivity* pede para mudar de lugar, isto é, que deseja ser transposta para o *FormularioActivity*. Feita essa leitura o *Android fecha a* `intent` *e instancia o* FormularioActivity*.

Ao colocar algo na `intent` , como ela vai para o *Android* e depois para o *FormularioActivity*, recuperamos o que penduramos. Portanto, vamos pendurar o `aluno` na `intent` e assim conseguiremos passar ele para o *FormularioActivity*.

Para isso, na linha de baixo de `Intent intentVaiProFormulario` , digitaremos `intentVaiProForulario` e inserimos um método, o `putExtra` . Dentro dos parênteses colocamos aquilo que desejamos que a nossa `intent` leve para o *FormularioActivity*. No caso, o `aluno` , e aproveitamos para adicionar uma identificação, uma espécie de etiqueta. Ficaremos com, `intentVaiProFormulario.putExtra("aluno", aluno)` . Ao todo teremos:

```
@Override
public void onItemCIick(AdapterView<?> lista, View item, int position, long id) {
    Aluno aluno = (Aluno) listaAlunos.getItemAtPosition(position);
    Intent intentVaiProFormuIario = new Intent(ListaAlunosActivity.this, FormularioActivity.cla
    intentVaiProFormulario.putExtra("aluno", aluno) startActivity(intentVaiProFormulario);
}
```

Perceba que o `aluno` que inserimos continua com problemas, pois o *Android* não consegue reconhecê-lo. Teremos que "serializar" esse `aluno`, isto é, transformá-lo em binário.

Para transformar a classe em algo que pode ser serializado, vamos na aba *Aluno.java*. Na mesma linha da `public class` e logo no topo da página, acrescentamos que a classe aluno implementa a *serializable*, teremos: `public class Aluno implements Serializable`. Vamos importar o `Serializable` através de um "Alt+Enter". Agora, o *java* saberá como converter as `String`, o `Long` e o `Double` em binários. Encontraremos nossa tela da seguinte maneira:

```
public class Aluno  implements Serializable {
    private Long id;
    private String nome;
    private String endereço;
    private String telefone;
    private String site;
    private Double nota;
}
```

Vamos voltar no *ListaAlunosActivity*. Repare que mais nada atrapalha o `aluno` de estar pendurado na `intent`. Fizemos a parte de enviar o `aluno` para o *FormularioActivity.java*. Vamos partir para a parte do quando os dados já estão lá.

Voltamos na aba `FormularioActivity.java`. No momento em que estamos criando nosso formulário, na `onCreate` se algo estiver pendurado na `intent` teremos que recuperar essa `intent` e popular isso na `OnCreate`. Para fazer isso vamos recuperar a `Intent`, digitando `Intent intent = getIntent`. Vamos importar isso através do "Alt+Enter". Agora, vamos recuperar o que está pendurado na `intent`, no caso, o `aluno`. Para isso, digitaremos `intent.getExtra` e como temos vários tipos de `getExtra`, usaremos o `Serializable`. Ficaremos com, `intent.getSerializableExtra`.

Como ainda pede um nome, vamos usar a identificação, a etiqueta usada anteriormente, e digitaremos `"aluno"`. Como o `getSerializableExtra` nos devolve o `aluno`, vamos falar que `Aluno aluno é = a intent.getSerializableExtra`. Damos um "Alt+Enter" para fazer o "Cast" do aluno.

Falta trazer o conteúdo do aluno e jogar na tela do formulário. Trazemos o aluno da `intent` que está no `ListaAlunos.Activity` e temos que lembrar de verificar se esse aluno é diferente de nulo e se ele for nulo, teremos que preencher o formulário. Para fazer essa verificação acrescentamos `if (aluno !=null)`.

Quando queríamos pegar os dados do formulário, utilizamos o `Helper`, por isso, vamos utilizá-lo novamente. Vamos dar um "Ctrl+C" e "Ctrl+V" na linha `helper = new FormularioHelper(this);` e deslocá-la para dentro das chaves, abaixo da linha do `setContentView`.

Abaixo da linha `if (aluno !=null)` digitamos `helper.preencheFormulario` e completamos com `aluno`. Ficaremos com, `helper.preencheFormulario(aluno)`. Vamos dar um "Alt+Enter" em cima do `preencheFormulario` e escolhemos "Create Method". O método será criado ao final da página `FormularioHelper.java`. Teremos na *FormularioActivity.java* o seguinte:

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

        helper = new FormularioHelper(this);

        Intent intent = getIntent();
        Aluno aluno = (Aluno)  intent.getSerializableExtra("aluno");
```

```
        if (aluno !=null) {
            helper.preencheFormulario(aluno);
        }

    }
```

Vamos acessar a aba *FormularioHelper.java*. Falta preencher os campos, basta acrescentar na linha seguinte o `campoNome.setText(aluno.getNome())` e os demais dados. Lembrando que a nota recebe um `setProgress` e também um `intValue`. Teremos:

```
public void preencheFormulario(Aluno aluno)  {
campoNome.setText(aluno.getNome());
campoEndereco.setText(aluno.getEndereco());
campoTelefone.setText(aluno.getTelefone());
campoSite.setText(aluno.getSite());
campoNota.setProgress(aluno.getNota().intValue());
}
```

Perceba que, os campos já foram todos introduzidos no `FormularioHelper`, isto é, estão prontos para uso, o que facilita muito as coisas!

Atenção para um detalhe!

Salvamos o nome, o telefone, o site e a nota. Falta salvar o `Id`, caso contrário, perderemos ele e isso complicará as coisas quando precisarmos da identificação para alterar o banco. Então, salvaremos o aluno inteiro, criando um atributo inteiro. Digitaremos no topo da página do `FormularioHelper.java`, após o `private final RatingBar campoNota`, um `private Aluno aluno`. No final do `FormularioHelper` criaremos, `aluno = new Aluno`. Agora, podemos apagar no `Aluno pegaAluno` a linha `Aluno aluno = new Aluno`, pois ele só precisa estar instanciado uma vez. Ficaremos no `FormularioHelper` com:

```
public FormularioHelper(FormularioActivity activity)  {
    campoNome = (EditText) activity.findViewById(R.id.formulario_nome);
    campoEndereço = (EditText) activity.findViewById(R.id.formulario_endereço);
    campoTelefone = (EditText) activity.findViewById(R.id.formulario_Telefone);
    campoNota = (RatingBar) activity.findViewById(R.id.formulario_nota);
    aluno = new Aluno();
}
```

Agora, no `preencheFormulario` vamos guardar o aluno inteiro no atributo. Digitaremos ao final dele `this.aluno =aluno`. Ficaremos com:

```
    public void preencheFormulario(Aluno aluno)  {
    campoNome.setText(aluno.getNome());
    campoEndereco.setText(aluno.getEndereco());
    campoTelefone.setText(aluno.getTelefone());
    campoSite.setText(aluno.getSite());
    campoNota.setProgress(aluno.getNota().intValue());
    this.aluno = aluno;
    }
```

Como salvamos o aluno no `preencheFormulario` é desse local que vai advir o `id`. Vamos verificar se isso está funcionando no código! Vamos salvar e rodar o emulador.

Conseguimos trazer para o formulário todas as informações de "Daniel", mas repare que se alterarmos e salvarmos o formulário estaremos apenas acrescentando mais "Daniels" na nossa lista. Esse problema resolveremos na sequência!