01

Recuperando dados do aluno

Transcrição

Nas aulas anteriores introduzimos novos botões! Agora, desejamos guardar de fato o nome dos alunos que são acrescentados.

Para isso, realizaremos alterações no arquivo, `FormularioActivity`.

Lembrando

O *Android* avisa qual item do menu é clicado através do `onOptionsItemSelected`. No `onOptionsItemSelected` acrescentamos também um caso e o `Toast` para que uma mensagem apareça quando o botão for clicado.

Para salvar os alunos recém adicionados, precisaremos recuperar os dados novos que inserimos. Então, vamos adicionar algumas informações na *Override* da `onOptionsItemSelected`. Temos:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
                case R.id.menu_formulario_ok;
                    Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH_SH(

                    EditText campoNome = (EditText) findViewById(R.Id.formulario_nome);
                    String nome = campoNome.getText().toString();

                    finish();
                    break;
    }

        return super.onOptionsItemSelected(item);
}
```

Damos um "Enter" e na linha abaixo de 'Toast' iremos acrescentar algumas informações. Como necessitamos armazenar os dados que introduzimos no formulário. A referência será buscada através do `findViewById`, portanto, utilizaremos entre parênteses a classe `R`, `.`, `Id` e o nome da tela, que é `formulario_nome` e indica quem está sendo procurado. Ao digitar a palavra "formulário", serão mostradas algumas opções. Ficaremos com, `findViewById(R.Id.formulario_nome)`.

Como esse componente é do tipo `EditText`, vamos adicionar `EditText` e o `campoNome`, ficaremos com, `EditText campoNome = findViewById(R.Id.formulario_nome)`.

Em cima do `campoNome`, para importar a classe, utilizaremos o atalho "Alt+Enter". O *Android* vai pedir para fazermos o *Cast*, então, damos um "Alt+Enter" de novo e pronto.

Ficaremos com `EditText campoNome = (EditText) findViewById(R.Id.formulario_nome)`. Primeiro, pegaremos o campo e agora pegaremos o valor dele acrescentando `campoNome`, `.` e `getText`. Adicionaremos uma `String` na frente disso, seguida de "nome" e um "=". Teremos, `String nome = campoNome.getText`.

Mas, se fizermos apenas isso ele devolverá um *editable* e não queremos isso. Então, adicionaremos depois do `getText` mais um `.` e `toString`. Faremos isso para converter o campo em algo que possa ser colocado de fato na `String`. Teremos, `String nome = campoNome.getText().toString()`. Ficaremos com :

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_formulario_ok;
            Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH_SHORT).shou

            EditText campoNome = (EditText) findViewById(R.Id.formulario_nome);
            String nome = campoNome.getText().toString();

            finish();
            break;
    }

    return super.onOptionsItemSelected(item);
}
```

Fizemos isso com o campo "nome" e faremos o mesmo com o campo endereço. Para isso, daremos um "Enter" depois do que acabamos de acrescentar e na próxima linha teremos que pegar novamente a referência do campo, no caso, o `Id` de endereço. Digitaremos, `findViewById(R.id.formulario_endereço)` e como também é um `EditText`, adicionaremos isso e `campo endereço`. Na sequência, importaremos e faremos o *Cast*. Por fim, teremos `EditText campoEndereço = campoEndereço findViewById(R.id.formulario_endereço)`. Na próxima linha adicionaremos o `String`, `endereco`, `=`, `campoEndereco`, o `getText`, que serve para pegar o texto e o `to String`. Teremos:

```
EditText campoEndereco = (EditText) findViewById(R.Id.formulario_Endereco);
String Endereco = campoEndereco.getText().toString();
```

Faremos o mesmo nos campos de telefone e site, basta selecionar o que acabamos de escrever, do `EditText` do nome até a `String` do Endereço, dar um "Comand+C" ou "Ctrl+C" e "Comand+V" ou "Ctrl+V". Colaremos isso abaixo do Endereço. Basta alterar os nomes pelos campos que desejamos, no caso "Site" e "Telefone". Teremos:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_formulario_ok;
            Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH_SHORT).shou
                EditText campoNome = (EditText) findViewById(R.Id.formulario_nome);
                String nome = campoNome.getText().toString();

                EditText campoEndereco = (EditText) findViewById(R.Id.formulario_Endereco);
                String Endereco = campoEndereco.getText().toString();

                EditText campoTelefone = (EditText) findViewById(R.Id.formulario_Telefone);
                String Telefone = campoTelefone.getText().toString();

                EditText campoSite = (EditText) findViewById(R.Id.formulario_Site);
                String Site = campoSite.getText().toString();
```

```
                finish();
                break;
        }

        return super.onOptionsItemSelected(item);
    }
```

Imagine se tivéssemos mantido o antigo botão de salvar, aquele que ficava na parte de baixo do formulário. Teríamos que repetir todo o código que acabamos de escrever para ele ter a mesma função. Nessas situações, quando é preciso repetir muitas vezes o mesmo código, podemos criar uma classe para que ela faça isso por nós. Portanto, vamos deslocar os códigos em uma classe auxiliar do formulário.

Para criar uma nova classe vamos no pacote "br.com.alura", clicamos com o botão direito e "New > Java Class". Vai aparecer uma janela com um campo *Name* para ser preenchido, nomearemos nossa classe de "FormularioHelper". Abrirá uma tela com a nova classe:

```
public class FormularioHelper {
}
```

Vamos fazer todos os `findViewById` que estão na aba `FormularioActivity.java` , na nova classe, justamente, para facilitar. Assim, não precisaremos ficar recriando eles sempre que necessário. Vamos introduzir os `findViewById` no início, no construtor do "FormularioHelper".

Para fazer o construtor basta dar um enter depois do `public class FormularioHelper` e ele já aparece na tela, basta escrever `public FormularioHelper` . Agora, dentro, iremos adicionar os `findViewById` .

Na próxima linha adicionaremos o `findViewById` e digitaremos o campo que queremos puxar, no caso, o `formulario_nome` . Ficaremos com, `findViewById(R.Id.formulario_nome)` . Repare que o `findViewById` ficará em vermelho, pois não é possível localizar o método.

O `findViewById` é um método oriundo da `AppCompatActivity` , e quando estamos dentro da `activity` podemos chamar o método normalmente. Mas, estamos no 'FormularioHelper.java'. Para utilizar o `findViewById` no 'FormularioHelper.java' necessitamos de uma referência da nossa `activity` . Vamos adicionar essa referência no construtor da nossa classe, adicionaremos entre os parênteses que seguem o `public formularioHelper` a referência de que estamos trazendo isso da `FormularioActivity` e nomearemos ela de "activity". Ficaremos com `FormularioHelper(FormularioActivity activity)` .

Como essa referência vem de fora do `FormularioHelper.java` temos que referenciar também o `findViewById` . Digitaremos na frente de tudo `activity` e um ponto, teremos, `activity.findViewById(R.Id.formulario_nome)` .

Falta acrescentar que é um `EditText` do `campoNome` que será `=` a `activity.findViewById(R.Id.formulario_nome)` . Teremos, `EditText campoNome = activity.findViewById(R.Id.formulario_nome)` . Damos um "Alt+Enter" para importar e fazemos também o *Cast*.

Vamos ter o seguinte:

```
public class FormularioHelper  {

    public FormularioHelper(FormularioActivity activity) {
```

```
        EditText campoNome = (EditText) activity.findViewById(R.Id.formulario_nome);
    }
}
```

Agora, temos o campo do nome pronto!

Basta reproduzir o que já escrevemos. Selecionamos com o mouse, 'EditText campoNome = (EditText) activity.findViewById(R.Id.formulario_nome)' e damos um "Command+C" e "Command+V". Agora, trocaremos apenas os nomes do `EditText` e alteraremos os `Id` .

Acrescentaremos um último campo, que é o da nota. Atenção aqui, ele não é um `EditView` , ele é um `RatingBar` ,então, temos que alterar os tipos das variáveis. Ficaremos com `RatingBar campoNota = (RatingBar) activity.findViewById(R.Id.formulario_nota)` . Não esquecendo de importar a classe com o "Alt+Enter".

E ficaremos com:

```
public class FormularioHelper {

    public FormularioHelper(FormularioActivity activity)  {
        EditText campoNome = (EditText) activity.findViewById(R.id.formulario_nome);
        EditText campoEndereco = (EditText) activity.findViewById(R.id.formulario_endereco);
        EditText campoTelefone = (EditText) activity.findViewById(R.id.formulario_telefone);
        EditText campoSite = (EditText) activity.findViewById(R.id.formulario_site);
        RatingBar campoNota = (RatingBar) activity.findViewById(R.id.formulario_nota);
    }
}
```

Agora, na aba `FormularioActivity.java` podemos apagar os `EditText` que tínhamos e apagaremos o seguinte:

```
EditText campoNome = (EditText) findViewById(R.Id.formulario_nome);
String nome = campoNome.getText().toString();
EditText campoEndereco = (EditText) findViewById(R.Id.formulario_Endereco);
String endereco = campoEndereco.getText().toString();
EditText campoTelefone = (EditText) findViewById(R.Id.formulario_Telefone);
String telefone = campoTelefone.getText().toString();
EditText campoSite = (EditText) findViewById(R.Id.formulario_Site);
String site = campoSite.getText().toString();
```

E ficaremos apenas com:

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item)  {
            switch (item.getItemId()) {
                case R.id.menu_formulario_ok;
                    Toast.makeText(FormularioActivity.this, "Aluno salvo!" , Toast.LENGTH_SHORT

                    finish();
                    break;
            }
```

```
        return super.onOptionsItemSelected(item);
    }
```

Agora, precisamos substituir. Quando formos buscar o item dentro da classe `ItemSelected`, iremos introduzir a classe do `FormularioHelper`. Vamos instanciar essa classe em algum lugar, por exemplo, no `onCreate`. Então, na `onCreate` adicionamos, `helper`, `=` e `new FormularioHelper`. Ficaremos com `helper = new FormularioHelper`. Só que ele ainda pede uma referência para a `activity`, então, digitaremos o `this` entre os parênteses, como parâmetro. Ficaremos com `FormularioHelper helper = new FormularioHelper (this)`.

Mas, adicionando no `Toast` o `helper`, ele não é encontrado, por isso, apagamos o `FormularioHelper` que tínhamos escrito e transformamos isso em atributo. Repare que o `helper` fica em vermelho, damos um "Alt+Enter" e escolhemos "Create final helper". O *Android* irá criar, acima da `Override` um `private FormularioHelper helper`. Teremos:

```
    private FormularioHelper helper;

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

        helper = new FormularioHelper (this);
    }
```

E quando quisermos utilizar o `helper`, vamos embaixo, na classe `onOptionsItemSelected` e depois do `Toast`, na próxima linha, acrescentamos `helper` e damos um `.`. Se não tivéssemos criado o atributo na `onCreate` não iríamos conseguir localizar o `helper` e ele continuaria em vermelho.

O que faremos a partir do `Helper` é pegar os dados dos aluno. Para isso, vamos adicionar uma `String`, teremos `String nome = helper.peganome` e faremos isso com todos os outros campos. O que não é muito prático.

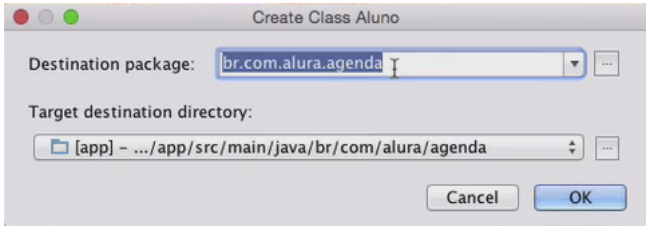
Através do `helper` desejamos sinalizar todos os campos que aparecem no formulário. Os campos da tela do celular, o nome, o endereço, a nota, o site e o telefone representam um "aluno" no sistema. O ideal é que tivéssemos um único método no `helper`, que seria `String nome = helper.pegaAluno`, responsável por trazer todos os dados que queremos. Mas, no lugar de `String` devemos ter um objeto de tipo `Aluno`. Ficaremos com `Aluno aluno = helper.pegaAluno`.

Teremos:

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
                case R.id.menu_formulario_ok;
                 Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH_SHORT)
                 Aluno aluno = helper.pegaAluno();

                 finish();
                 break;
        }

        return super.onOptionsItemSelected(item);
    }
```

Vamos fazer um "Alt+Enter" no objeto `Aluno`. O *Android Studio* sugere que criemos a classe alunos, a "Create class `Aluno`". Portanto, vamos selecionar essa opção e criar a classe.

Abrirá uma janela, que tem um campo `Destination package`. Vamos criar um pacote que já conhecemos e chamá-lo de modelo, o `br.com.alura.agenda.modelo`. Damos um "Ok".

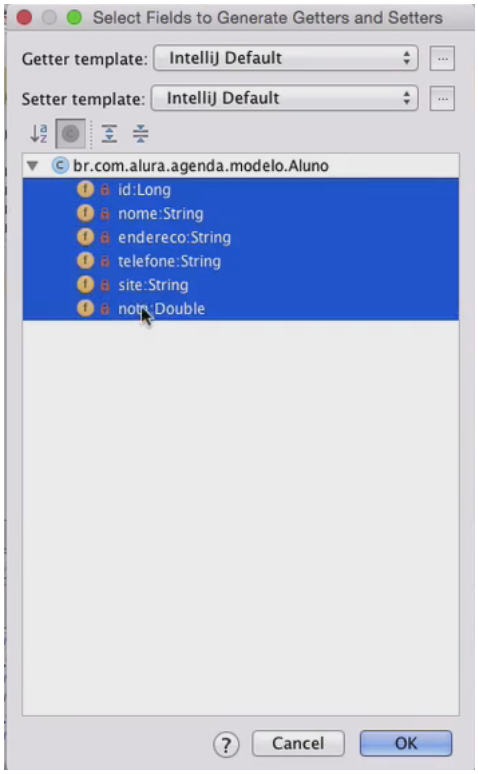


Ele criará uma pasta chamada "modelo" e ela estará junto com as outras, no lado direito da tela. Nela teremos a classe "Aluno" e dentro do 'Aluno.java' colocaremos os dados do aluno.

Vamos acrescentar abaixo do `public class`, na próxima linha, o `private String nome` e faremos o mesmo com os demais campos que queremos trazer para o `Aluno.java`. A nota, entretanto, será `Double` e não `String`. Logo menos, começaremos a trabalhar com o banco de dados, vamos acrescentar também um `Id` para esse aluno e já deixar isso para o futuro. Logo na linha de baixo de `public class` digitamos `private Long id`. Teremos:

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

Como colocamos que todos os nossos objetos estão privados, teremos que criar um `Getter and Setter`. Damos uns dois "Enter" depois da última linha e no *Mac* utilizamos o atalho "Command+N" e no *Windows* "Alt+Insert" e selecionamos o "Getter and Setter". Vai abrir um janela e nela selecionaremos todos os campos que gostaria que ele gerasse um "Getter" e um "Setter" e damos um "Ok".



Automaticamente nossa classe terá vários "Getter" e "Setter" definidos.

Agora que a classe "Aluno" está certa, vamos voltar no `FormularioActivity`. No `Helper` acrescentamos o `pegaAluno`, mas ele ainda está em vermelho, falta acrescentar um método para ele. Para isso, damos um "Alt+Enter" em cima do `pegaAluno` e selecionamos "Create method 'pegaAluno'". Pronto, agora temos um `public Aluno pegaAluno`. Ficaremos com:

```
public class FormularioHelper  {

    public FormularioHelper(FormularioActivity activity)   {
        EditText campoNome = (EditText) activity.findViewById(R.Id.formulario_nome);
        EditText campoEndereco = (EditText) activity.findViewById(R.Id.formulario_endereco);
        EditText campoTelefone = (EditText) activity.findViewById(R.Id.formulario_telefone);
        EditText campoSite = (EditText) activity.findViewById(R.Id.formulario_site);
        RatingBar campoNota = (RatingBar) activity.findViewById(R.Id.formulario_nota);
    }

    public Aluno pegaAluno()   {
    return null;
    }
}
```

Podemos introduzir abaixo do método `pegaAluno` um aluno, assim, acrescentamos o `Aluno aluno = new Aluno`. E na linha de baixo digitamos `aluno.setNome`. O nome do aluno vem do formulário, então, acrescentamos `campoNome` , `.getText`. Ficaremos com `aluno.setNome(campoNome.getText)`.

Mas, o campo nome ficará em vermelho, isso ocorre porque antes declaramos que ele era um `EditText`. Para resolver esse problema basta apagar o `EditText` de todos os campos da classe `FormularioHelper` e deixar apenas o uso deles:

```
public class FormularioHelper  {

    public class FormularioHelper(FormularioActivity activity)  {
```

```
            campoNome = (EditText) activity.findViewById(R.Id.formulario_nome);
            campoEndereco = (EditText) activity.findViewById(R.Id.formulario_endereco);
            campoTelefone = (EditText) activity.findViewById(R.Id.formulario_telefone);
            campoSite = (EditText) activity.findViewById(R.Id.formulario_site);
            campoNota = (RatingBar) activity.findViewById(R.Id.formulario_nota);
    }

    public Aluno pegaAluno()   {
            Aluno aluno = new Aluno ();
            aluno.setNome(campoNome.getText());

            return null;
    }
}
```

Agora, nenhum deles está definido. Então, em todos os campos faremos "Alt+Enter" e criaremos "Fields". Todos os campos estarão com atributos:

```
public class FormularioHelper  {

    private EditText campoNome;
    private EditText campoEndereco;
    private EditText campoTelefone;
    private EditText campoSite;
    private Ratingbar campoNota;

    public class FormularioHelper(FormularioActivity activity)  {
        campoNome = (EditText) activity.findViewById(R.Id.formulario_nome);
        campoEndereco = (EditText) activity.findViewById(R.Id.formulario_endereco);
        campoTelefone = (EditText) activity.findViewById(R.Id.formulario_telefone);
        campoSite = (EditText) activity.findViewById(R.Id.formulario_site);
        campoNota = (RatingBar) activity.findViewById(R.Id.formulario_nota);
    }

    public Aluno pegaAluno() {
        Aluno aluno = new Aluno ();
        aluno.setNome(campoNome.getText());

        return null;
    }
}
```

Agora conseguimos acessar a classe 'pegaAluno'!

Já digitamos `aluno.setNome(campoNome.getText)` e como ele nós devolve um *editable*, devemos converter isso. Usaremos para tanto um `toString` . Não esqueça de fechar com um `;` . Vamos repetir o `aluno.setNome(campoNome.setText().toString())` por quatro vezes para inserir também os outros campos. Alteramos os nomes do campo inserindo o que ainda falta: endereço, telefone, site e nota.

A nota é um pouco diferente, para descrevê-la acrescentamos um `getProgress` que devolverá um `Double` e nele escreveremos `valueOf` . Para a nota teremos: `aluno.setNota(Double.valueOf(campoNota.getProgress()))` . Por fim, devolvemos o aluno que acabamos de construir com `return aluno` . Teremos:

```
public Aluno pegaAluno()  {
    Aluno aluno = new Aluno ();
    aluno.setNome(campoNome.getText().toString());
    aluno.setEndereco(campoEndereco.getText().toString());
    aluno.setTelefone(campoTelefone.getText().toString());
    aluno.setSite(campoSite.getText().toString());
   aluno.setNota(Double.valueOf(campoNota.getProgress()));
    return aluno;
}
```

Vamos voltar ao `FormularioActivity.java` . Observe onde estávamos:

```
@Override
public boolean onOptionsItemSelected(MenuItem item)  {
    switch (item.getItemId()) {
        case R.id.menu_formulario_ok;
            Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH_SHORT).shou
            Aluno aluno = helper.pegaAluno();

            finish();
            break;
    }

    return super.onOptionsItemSelected(item);
}
```

Para confirmar se o código está correto vamos deslocar o `Toast.makeText(FormularioActivity.this, "Aluno salvo!"` para a linha de baixo do `Aluno aluno = helper.pegaAluno` , através do "Command+V" e "Command+C". No `Toast` concatearemos, depois de "Aluno", o nome. Digitaremos:

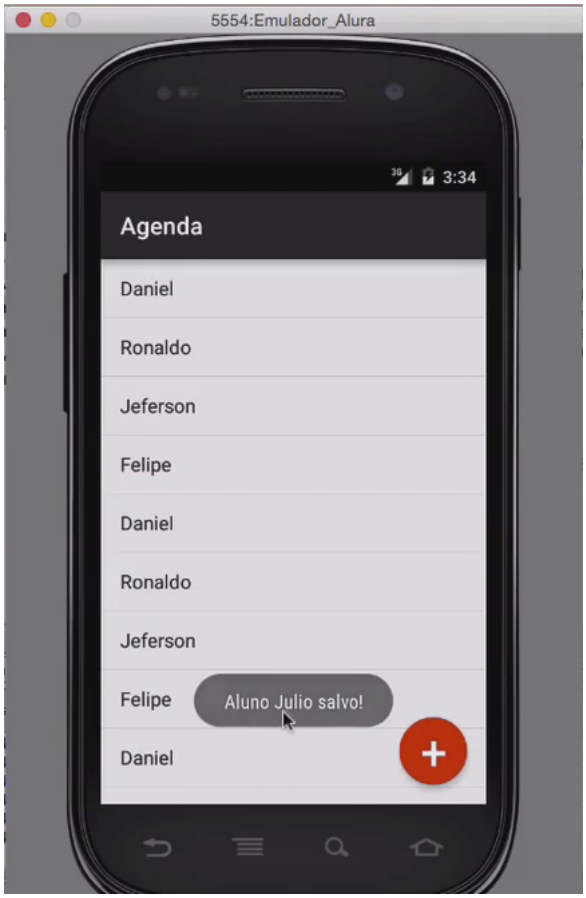
```
Toast.makeText(FormularioActivity.this. "Aluno" + aluno.getNome() + "salvo!", Toast.LENGTH_SHOR
```

E no todo teremos:

```
@Override
public boolean onOptionsItemSelected(MenuItem item)  {
    switch (item.getItemId()) {
        case R.id.menu_formulario_ok;

            Aluno aluno = helper.pegaAluno();
            Toast.makeText(FormularioActivity.this, "Aluno" + aluno.getNome() + "salvo!", To

            finish();
            break;
    }

    return super.onOptionsItemSelected(item);
}
```

Vamos rodar o emulador! No campo formulário acrescentamos um novo aluno, o Júlio:

Agora, quando clicarmos no "Salvar", deve aparecer o "Aluno Júlio salvo".



Conseguimos preparar o caminho para salvar o nome do aluno no banco de dados!