02

Fragments e o pacote de compatibilidade

Muitas pessoas não se contentam com um simples *smartphone* por causa do tamanho da tela. Uma tela maior significa a possibilidade de aplicações com telas mais ricas, apresentando uma quantidade maior de recursos ou opções. Devido a esses e outros fatores a venda de tablets vem tomando proporções cada vez mais expressivas.

Para os desenvolvedores, entretanto, o aumento de funcionalidades da tela traz uma dificuldade: como fazer a implementação de uma tela complexa sem aumentar demasiadamente o número de linhas da *activity* relacionada a ela? O ideal seria separar a parte da tela relacionada a uma certa funcionalidade, criar uma classe e fazer a *activity* delegar para ela a responsabilidade de lidar com os eventos daquela porção da tela.

Percebendo essa necessidade de "quebrar" a tela em pedaços menores, os criadores do Android resolveram criar a **API de *fragments*** que facilita esse processo de fragmentação da *view*. Essa novidade está disponível a partir da versão *HoneyComb* (Android 3.0).

Fragments

Os *fragments* são pedaços de tela do usuário numa *activity*. Eles apareceram na **API *level* 11** ou seja, na distribuição do *Honeycomb*. Foram desenvolvidos para rodar nos *tablets*, desde que com a versão correta do Android. (Lembre-se que alguns *tablets* são lançados com versões de *smartphone*).

Desde sua criação, os *fragments* têm sido utilizados pelos desenvolvedores para tornar o *layout* das telas mais flexível. Em toda aplicação, desde o momento do desenvolvimento, devemos levar em conta a usabilidade e sabemos que o usuário pode optar por operar seu aparelho de pé (*portrait*) ou deitado (*landscape*). Em uma tela grande o impacto da orientação da tela na usabilidade é ainda maior, de forma que vamos optar por criar dois *layouts*, um para cada caso.

A idéia de trabalhar com *fragments* parece simplificar e organizar o desenvolvimento de *views* complexas. Infelizmente só seria possível implementar uma solução usando esse e outros recursos da *Honeycomb* em dispositivos novos.

Como faríamos para desenvolver uma aplicação compatível com *smartphones* mais antigos e que pudessem desfrutar do espaço extra dos *tablets*? Teríamos que escrever duas aplicações distintas? Uma usando *fragments* e outra apenas *activities*?

Para resolver esse problema foi criado um **pacote de compatibilidade**: um **jar** que encontra-se em nossa **SDK**, que possui a classe `FragmentActivity` que nos fornece a capacidade de utilizar *fragments*, já que a versão antiga da classe *activity* não possui o método `getFragmentManager()`.

Utilizando esse pacote de compatibilidade somos capazes de desenvolver aplicações que rodam em dispositivos antigos e quando abertas em um *tablet* aproveite devidamente o espaço extra do aparelho.

Listagem de Provas

Vamos preparar nossa aplicação para mostrar uma listagem das provas agendadas, para isso começaremos criando nosso novo modelo, a classe `Prova`:

```
public class Prova implements Serializable {
    private String data;
    private String materia;
    private List<String> topicos = new ArrayList<String>();

    //getters e setters
}
```

Precisamos agora gerar a listagem de provas, para isso criaremos uma *view* na pasta *res/layout* que contenha uma `ListView` que vamos inflar com as provas. Vamos chamar de **provas_lista.xml**:

```
<?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" >
```

```
    <ListView
        android:id="@+id_provas_lista/list_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</LinearLayout>
```

Queremos uma listagem de todas as provas. Como mostrar todas informações de todas as provas, inclusive os tópicos da prova na mesma listagem?

Para evitar criar uma tela com muita informação, vamos quebrar a visualização das provas em duas telas: uma com a listagem e outra com os detalhes da prova selecionada.

Vamos diferenciar um pouco mais o *layout* da visualização dos detalhes criando uma borda e um fundo de cor diferenciada. Para isso vamos criar um arquivo para ser usado como fundo na pasta ***res/drawable-nodpi*** com o *layout* da borda:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <stroke android:width="5dp" android:color="#D4D4D4"/>
    <solid android:color="#6B6B6B" />
    <padding
        android:left="15dp" android:top="15dp"
        android:right="15dp" android:bottom="15dp"/>
    <corners android:radius="15dp" />
</shape>
```

Agora vamos precisar criar as *activities* para controlar as views da listagem e dos detalhes das provas. O processo seria idêntico ao que já fizemos para o cadastro e listagem de alunos. Vamos agora nos preocupar com um problema: Em um *tablet* na posição deitada (*landscape*) existe espaço suficiente na tela para para acomodar tanto a listagem das provas quando os detalhes da prova selecionada. Vamos desenvolver essa nova funcionalidade pensando no funcionamento tanto em *smartphones* quanto em *tablets* com o uso de *fragments*.

Para isso vamos precisar copiar do diretório de instalação de nossa SDK o **jar de compatibilidade**.

Agora vamos criar o *fragment* que estará associado à *view* de listagem das provas:

```
public class ListaProvasFragment extends Fragment{
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {

        View layoutProvas = inflater.inflate(R.layout.provas_lista,
            container, false);

        return layoutProvas;
    }
}
```

Vamos criar agora o *layout* para dispositivos de tela pequena, que serão apenas capazes de mostrar ou a listagem ou os detalhes. Para isso vamos criar na pasta *res/layout* a *view* **provas.xml**:

```
<?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" >

        <FrameLayout android:id="@+id/provas_view"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
</LinearLayout>
```

Sabemos que o controle da tela ainda é feito por *activities*. Vamos criar a *activity* que irá manipular os *fragments* que criamos. Para que tenhamos acesso ao `FragmentManager` precisamos extender de `FragmentActivity`:

```
public class ProvasActivity extends FragmentActivity{
    @Override
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);

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

```
    }
}
```

Vamos agora, no `onCreate` de nossa `ProvasActivity` vincular o *fragment* da listagem ao `<FrameLayout>`:

```
FragmentTransaction transaction =
    getSupportFragmentManager().beginTransaction();

transaction.replace(R.id.provas_view,
    new ListaProvasFragment(), "listaProvas");

transaction.commit();
```

Com a estrutura montada até o momento podemos popular os dados em nosso `ListaProvasFragment`. Vamos buscar o `ListView` e inflá-lo com os dados de uma prova:

```
Prova prova1 = new Prova("20/03/2012", "Matematica");
prova1.setTopicos(Arrays.asList("Algebra linear", "Integral", "Diferencial"));

Prova prova2 = new Prova("25/03/2012", "Português");
prova2.setTopicos(Arrays.asList("Complemento nominal", "Orações Subordinadas"));

List<Prova> provas = Arrays.asList(prova1, prova2);

this.listViewProvas =
        (ListView) layoutProvas.findViewById(R.id_provas_lista.list_view);
this.listViewProvas.setAdapter(new ArrayAdapter<Prova>(getActivity(),
            android.R.layout.simple_list_item_1, provas));
```

Nesse primeiro momento não há nenhum ganho aparente. Para sermos capazes de colocar a listagem de provas na tela criamos dois arquivos de *layout* (o **provas.xml** e o **provas_lista.xml**) a duas classes (a `ProvasActivity` e o `ListaProvasFragment`). Se não optássemos por usar *fragments* seriamos capazes de realizar a mesma tarefa criando apenas uma *activity* e sua respectiva *view* exatamente como fizemos na listagem de alunos. A vantagem do uso de *fragments* só aparecerá mais tarde quando quisermos dar suporte a um comportamento diferenciado para dispositivos de tela grande.

Criando os detalhes da prova selecionada

Vamos preparar a visualização dos detalhes da prova para que possamos mais tarde clicar em uma das provas da listagem e então ver seus detalhes.

Vamos começar criando a *view* de detalhes, usando o estilo da borda como *background*:

Arquivo ***res/drawable-nodpi/border.xml***:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <!--     Cria linha com bordas arredondadas e tom de cinza    -->
    <stroke android:width="5dp" android:color="#D4D4D4"/>
    <corners android:radius="15dp" />

    <!--     Escolhe a cor de fundo        -->
    <solid android:color="#6B6B6B" />

    <!--     Espaço entre conteudo e borda     -->
    <padding
        android:left="15dp" android:top="15dp"
        android:right="15dp" android:bottom="15dp"/>
</shape>
```

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:layout_margin="20dp" android:background="@drawable/border">

    <TableRow >
        <TextView
            android:layout_width="match_parent" android:layout_height="wrap_content"
            android:textStyle="bold" android:textSize="25sp"
            android:text="DETALHES DA PROVA"/>
    </TableRow>
```

```
        <TableRow >
            <TextView
                android:layout_width="wrap_content" android:layout_height="wrap_content"
                android:textStyle="bold" android:text="Materia:"/>

            <TextView
                android:id="@+id/detalhe_prova_materia"
                android:layout_width="match_parent"  android:layout_height="wrap_content" />
        </TableRow>

        <!-- OUTRAS INFORMACOES DA PROVA -->

        <TableRow >
            <ListView
                android:id="@+id/detalhe_prova_topicos"
                android:layout_width="match_parent"  android:layout_height="match_parent"/>
        </TableRow>
    </TableLayout>


    public class DetalhesProvaFragment extends Fragment{
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {

            View layoutDetalhes = inflater.inflate(R.layout.provas_detalhe,
                container, false);

            return layoutDetalhes;
        }
    }
```

Agora basta implementarmos a apresentação dos dados na *view* com o `DetalhesProvaFragment` criando o método `selecionaProva` que vai substituir a *view* apresentada na tela (atualmente a listagem de provas) pela *view* de detalhes da prova. Para que isso seja possível devemos alterar o *fragment* associado ao *FrameLayout* da *view* que está sendo apresentada na tela, que é a **provas.xml**.

```
    <LinearLayout ...>
        <FrameLayout android:id="@+id/provas_view" .../>

         <!-- vamos inflar agora com o Fragment DetalhesProva -->
    </LinearLayout>
```