UI e Fragments

Activities e Fragments

Activity

Representam uma entidade de interação com o usuário (uma atividade);

Possuem um conteúdo de referencia a ser exibido, definido pelo método setContentView();

Descendem da classe Activity;

Possui um ciclo de vida característico;

Fragment

Intuitivamente descritos como “fragmentos” de tela, são componentes independentes;

Tem seu próprio ciclo de vida;

Encapsulam funcionalidades que os tornam mais fáceis de reusar dentro de Activities e Layouts;

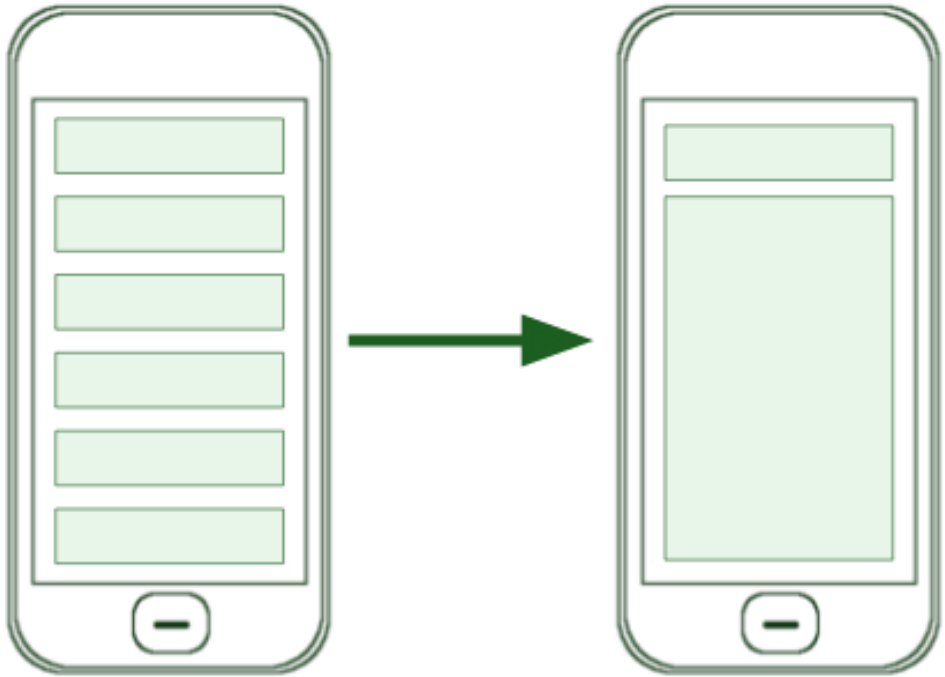
Introduzidos na versão Honneycomb

Fluxo Master/Detail

Imaginem a seguinte situação:

O aplicativo tem uma Activity de lista de itens e, quando se clica em um deles, abre-se uma outra Activity com os detalhes.

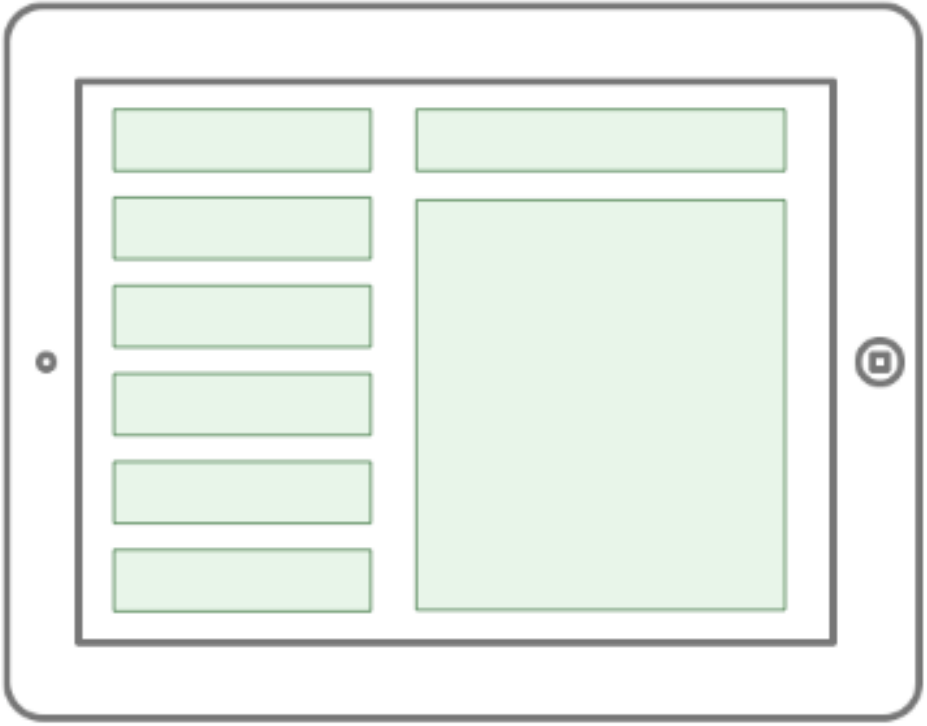
Esse tipo de fluxo de telas é chamado de Master/Detail, levando da listagem de itens até o detalhe do item selecionado.



Fluxo Master/Detail

No caso de um tablet, tanto a lista de itens quanto os seus detalhes podem ser apresentados na mesma Activity.

Agora ao clicar em um item da lista, a área da direita é substituída por outra, ao invés de iniciar uma nova, como foi feito no exemplo do celular.



Fragments

Para criar fragmentos você precisa estender da classe **Fragment** ou de suas derivadas (ListFragment, DialogFragment ou PreferenceFragmentCompact).

Um fragmento é executado no contexto da Activity, isto é, uma Activity incorpora um Fragmento.

Seus eventos básicos (ciclo de vida) associados ao Fragments são semelhantes aos das Activities, porém com certas diferenças:

Fragments: métodos de callback

onCreateView():

Similar ao onCreate() da Activity, é neste métodos onde a interface gráfica é criada ou inflada.

onDestroyView():

Equivale ao onDestroy() da Activity e é chamado imediatamente antes do Fragment ser destruído.

Opera independentemente da Activity que incorpora o fragmento.

Usualmente utilizado para liberar recursos (ex: cursores de dados)

onAttach():

Pode-se obter referência da Activity pai.

onDetach():

Última tarefa a ser realizada, mesmo que tecnicamente o fragmento tenha sido destruído.

onActivityCreated():

O método onCreate do Activity foi finalizado e é seguro interagir com a UI

Adicionando fragmentos ao seu layout ...

De forma estática:

Introduz-se o fragmento ao layout através da tag <fragment>, tal qual um outro componente qualquer.

Esta forma é particularmente útil se as interfaces compartilham componentes comuns em sua construção, evitando que haja repetição do código XML que descreve a UI.

```
<LinearLayout xmlns:android="..."
                    xmlns:tools="...">
    <fragment
      android:id="@+id/mainFragment"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      class="br.unip.tap.MainFragment"
      tools:layout="@layout/main_fragment">
    </fragment>
</LinearLayout>
```

Adicionando fragmentos ao seu layout ...

Usando o FragmentManager:

A classe FragmentManager permite adicionar, remover e substituir fragmentos dinamicamente no layout da Activity.

É acessado na Activity por meio do método getSupportFragmentManager()

Toda manipulação dinâmica envolvendo fragmentos deve ser feita utilizando transações através da classe FragmentTransaction.

```
<LinearLayout xmlns:android="..."
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
    <FrameLayout
      android:id="@+id/fragment_content"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
    />
</LinearLayout>
```

Adicionando fragmentos ao seu layout ...

```
// Acessa o FragmentManager
FragmentManager fm =
            getSupportFragmentManager();

// Incia uma transação e adiciona o fragmento
FragmentTransaction ft =
            fm.beginTransaction();
ft.add(R.id.fragment_content,
            new MainFragment());
ft.commit();
```

```
// Substitui um fragmento
FragmentTransaction ft = fm.beginTransaction();
ft.replace(R.id.fragment_content,
            new MainFragment());
ft.commit();

// Remove um Fragment
Fragment fragment =
        fm.findFragmentById(R.id.fragment_content);
FragmentTransaction ft = fm.beginTransaction();
ft.remove(fragment);
ft.commit();
```

Informando parâmetros para um Fragment

Uma **Activity** pode utilizar um objeto da classe **Bundle** para passar informações e parâmetros para o **Fragment**.

Dentro do **Fragment** podemos pegar essas informções no método **onActivityCreated**.

```
MainFragment mainFragment = new MainFragment();
// Passando um link
Bundle bundle = new Bundle();
bundle.putString("link", link);
mainFragment.setArguments(bundle);


public void onActivityCreated
                        (Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    Bundle bundle = getArguments();
    if (bundle != null)
        String link = bundle.getString("link");
}
```

Comunicação entre Fragments

Para diminuir o acoplamento de código e aumentar a capacidade de reutilização dos Fragments nunca os faça se comunicarem diretamente entre si.

Toda comunicação deve ser feita através da Activity que o incorpora. Para isso, um **Fragment deve definir uma interface** interna e a **Activity que o usa deve implementá-la**.

Como garantia de compatibilidade, no método **onAttach**(), verifique se a Activity implementa corretamente essa Interface.

Comunicação entre Fragments (exemplo)

```
// Definição de um Fragment com interface:
public class MainFragment extends Fragment {
   private OnItemSelectedListener listener;

   public interface OnItemSelectedListener {
      public void onItemSelected(String link);
   }
   @Override
   public void onAttach(Context context) {
      super.onAttach(context);
      if (context instanceof OnItemSelectedListener)
         listener = (OnItemSelectedListener) context;
      else
         throw new ClassCastException();
   }
}
```

```
// No Activity pai faz-se a implementação da interface:
public class MainACtivity extends Activity
implements MainFragment.OnItemSelectedListener{
      public void onItemSelected(String link) {
         // Código que manipula componentes e o Fragment
      }
}

// No Fragment podemos utilizar o método
onItemSelected() implementado pela Activity:
public void updateDetail(String link) {
      listener.onItemSelected(link);
}
```