Programação para Dispositivos Móveis

Aula 21 - Fragmentos

Ana Karina D. Salina de Oliveira

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul

Programação para Dispositivos Móveis
2018

Roteiro

1. Fragmentos
2. Criação de um fragmento
3. Adição de um fragmento
4. Gerenciamento de Fragmentos
5. Ciclo de Vida dos Fragmentos

`https://developer.android.com/guide/components/fragments.html?hl=pt-br`

Android
Fragmentos

- Um Fragment representa o comportamento ou uma parte da interface do usuário em uma Activity.
- É possível combinar vários fragmentos em uma única atividade
  - para compilar uma IU de vários painéis
  - e reutilizar um fragmento em diversas atividades.
- Um fragmento é como uma seção modular de uma atividade,
  - que tem o próprio ciclo de vida,
  - recebe os próprios eventos de entrada,
  - pode ser adicionado ou removido com a atividade em execução (uma espécie de "subatividade"que pode ser reutilizada em diferentes atividades).

Android
Fragmentos

- Um fragmento deve sempre ser embutido em uma atividade
- O ciclo de vida de um fragmento é diretamente impactado pelo ciclo de vida da atividade do host.
  - quando a atividade é pausada, todos os fragmentos também são
  - quando a atividade é destruída, todos os fragmentos também são.

- Cada fragmento é processado de forma independentemente, como adicionar ou removê-los.
  - os fragmentos são adicionados em pilha de retorno da atividade
  - cada entrada da pilha de retorno é um registro da transação de fragmento que ocorreu.
  - a pilha de retorno permite que o usuário inverta uma transação de fragmento (navegue para trás), pressionando o botão Voltar.

Android
Fragmentos

- É possível inserir um fragmento no layout,
  - declarando-o no arquivo de layout da atividade, como um elemento <fragment>
  - ou no código do aplicativo adicionando-o a um ViewGroup existente.
- No entanto, não é necessário que um fragmento faça parte do layout da atividade;
  - é possível usar um fragmento sem a IU como um trabalhador invisível da atividade.

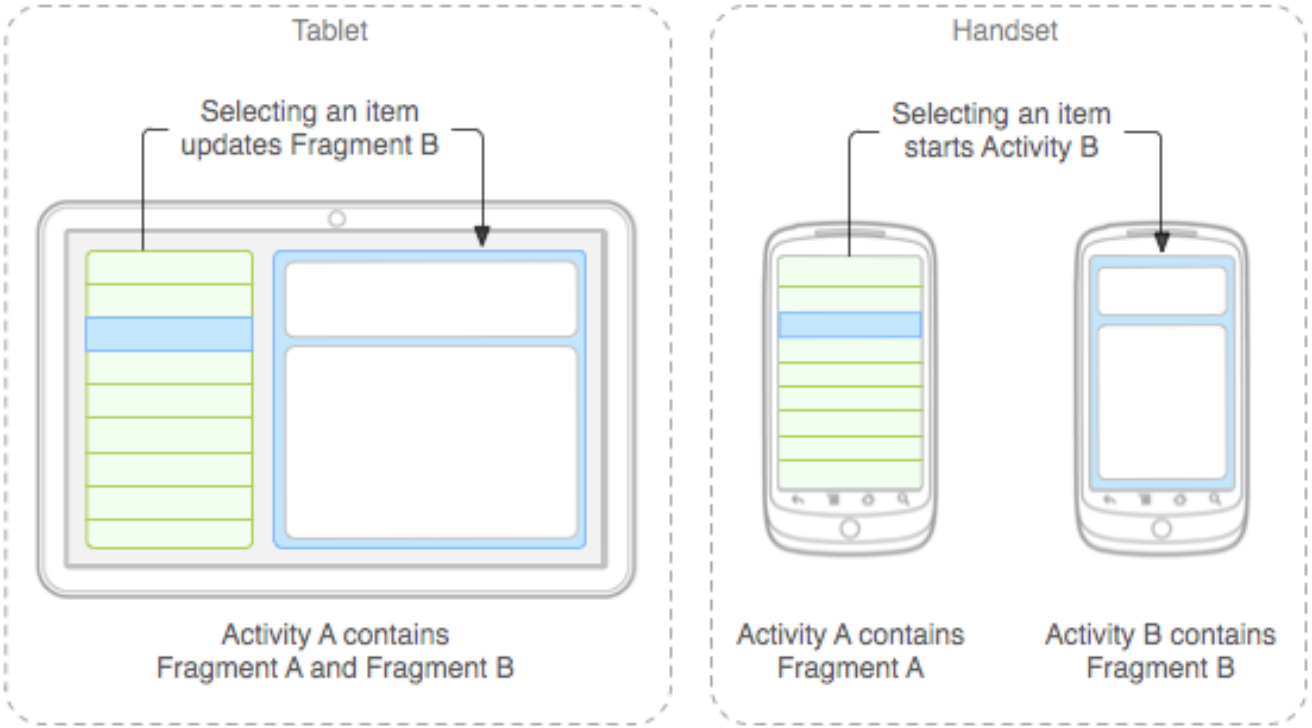
Android
Fragmentos

- O Android introduziu os fragmentos no Android 3.0 (API de nível 11)
  - para suportar mais projetos de telas grandes com IU flexíveis e dinâmicas, como tablets.
- Como a tela de um tablet é muito maior que a de um celular, há mais espaço para combinar e alternar componentes da IU.
- Os fragmentos permitem tais projetos sem haver a necessidade de gerenciar alterações complexas na hierarquia de exibições.
- Ao dividir o layout de uma atividade em fragmentos, é possível modificar a aparência da atividade em tempo de execução.

Android
Fragmentos

- Cada fragmento deve ser projetado como um componente modular e reutilizável da atividade.
- Cada fragmento define o próprio layout e comportamento com os próprios retornos de chamada do ciclo de vida,
  - permitindo incluir um fragmento em várias atividades para poder projetá-lo para reutilização
  - permitindo alterar as combinações de fragmentos para tamanhos de tela diferentes.
- Ao projetar o aplicativo para ser compatível com tablets e celulares,
  - você poderá reutilizar os fragmentos em diferentes configurações de layout com base no espaço de tela disponível.

Exemplo de como dois módulos de IU definidos pelos fragmentos podem ser combinados no tablet e no celular.



Criação de um fragmento

- Para criar um fragmento, é preciso criar uma subclasse de Fragment (ou uma subclasse existente dela).
- A classe Fragment tem um código que é muito parecido com o de uma Activity.
  - contém métodos de retorno de chamada semelhantes aos de uma atividade, como onCreate(), onStart(), onPause() e onStop().
- Para converter um aplicativo do Android existente para usar os fragmentos,
  - basta mover o código dos métodos de chamada da atividade para os respectivos métodos de chamada do fragmento.

- Geralmente, deve-se implementar pelo menos os seguintes métodos de ciclo de vida:
  - onCreate() $\rightarrow$ chamado ao criar o fragmento
  - onCreateView() $\rightarrow$ chamado no momento que o fragmento desenhar a interface do usuário pela primeira vez.
  - onPause() $\rightarrow$ chamado com o primeiro indício de que o usuário está saindo do fragmento.

Adição de uma interface do usuário

- Para fornecer um layout para um fragmento, deve-se implementar o método onCreateView(),
  - que será chamado no momento em que o fragmento deve desenhar o layout.
- A implementação desse método deve retornar uma View, que é a raiz do layout do fragmento.
- Se o fragmento for uma subclasse de ListFragment, a implementação padrão retornará uma ListView de onCreateView()
- Para retornar um layout de onCreateView(), é possível inflá-lo a partir do layout definido no XML.
  - O método onCreateView() fornece um objeto LayoutInflater.

Adição de uma interface do usuário

```
public static class ExampleFragment extends Fragment{
  @Override
  public View onCreateView(LayoutInflater inflater,
       ViewGroup container, Bundle savedInstanceState){
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.example_fragment,
       container, false);
  }
}
```

- O parâmetro container é o pai de ViewGroup (do layout da atividade) onde o fragmento será inserido.
- O parâmetro savedInstanceState é um Bundle que fornece dados sobre a instância anterior do fragmento quando este está sendo retomado (restaurado seu estado).

Android
Fragmentos

```
inflate(R.layout.example_fragment, container, false);
```

- O método inflate() usa três argumentos:
  - O ID de recurso do layout que você quer inflar.
  - O ViewGroup que será pai do layout inflado.
  - Um booleano que indica se o layout inflado deve ser anexado a ViewGroup (o segundo parâmetro) durante a inflação. (nesse caso, isso é falso, pois o sistema já está inserindo o layout inflado no container — retornar como verdadeiro criaria um grupo de exibições redundante no layout final).

Adição de um fragmento a uma atividade

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ArticleReaderFragment"
            android:id="@+id/viewer"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

- Há três maneiras de fornecer um ID para um fragmento que é utilizado para restaurar um fragmento:
  - especificando um código exclusivo ao **android:id**
  - especificando uma string exclusiva ao atributo **android:tag**
  - em caso de não fornecer nenhuma das alternativas anteriores, o sistema usará o código da exibição do contêiner.

Adicionando o fragmento pelo programa em execução

- É possível adicionar fragmentos ao layout da atividade enquanto a atividade está em execução,
- Basta especificar um ViewGroup no qual posicionar o fragmento.
- Para realizar transações de fragmentos na atividade (como adicionar, remover ou substituir um fragmento), você deve usar APIs de FragmentTransaction.

```
FragmentManager fM = getFragmentManager();
FragmentTransaction fragTransact=fM.beginTransaction();
```

Adicionando o fragmento pelo programa em execução

- É possível adicionar um fragmento usando o método add()
- O primeiro argumento passado para add() é ViewGroup, onde o fragmento deve ser colocado, especificado pelo ID de recurso
- o segundo parâmetro é o fragmento a ser adicionado.
- Ao realizar as alterações com FragmentTransaction, deve-se chamar commit() para que as alterações entrem em vigor.

```
ExampleFragment fragment = new ExampleFragment();
fragTransact.add(R.id.fragment_container, fragment);
fragTransact.commit();
```

Adição de um fragmento sem IU

- É possível usar um fragmento para fornecer o comportamento de segundo plano para a atividade sem apresentar IU adicional.
- Para adicionar um fragmento sem uma IU, adicione o fragmento da atividade usando add(Fragment, String)
  - fornecendo uma "tag"de string exclusiva para o fragmento, em vez de um código de exibição.
  - Isso adiciona o fragmento, mas, como ele não está associado a nenhuma exibição no layout da atividade, não recebe chamadas de onCreateView(). Portanto, não é necessário implementar esse método.

Gerenciamento de fragmentos

- Para gerenciar os fragmentos na atividade, você precisa usar FragmentManager.
- Para adquiri-lo, chame getFragmentManager() na atividade.
- Algumas funções do FragmentManager incluem:
  - Adquirir fragmentos existentes na atividade, com findFragmentById() ou findFragmentByTag().
  - Retirar os fragmentos da pilha de retorno com popBackStack().
  - Registrar uma escuta para as alterações na pilha de retorno com addOnBackStackChangedListener().

Realização de transações com fragmentos

- Um grande recurso fornecido por fragmentos em atividades é a possibilidade de adicionar, remover, substituir e realizar outras ações com eles.
- Cada transação é um conjunto de alterações que você deseja realizar ao mesmo tempo.
- É possível definir todas as alterações desejadas para uma transação usando métodos como add(), remove() e replace().
- Em seguida, para aplicar a transação à atividade, chame commit().

Realização de transações com fragmentos

- É possível adquirir uma instância de FragmentTransaction do FragmentManager da seguinte forma:

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction =
  fragmentManager.beginTransaction();
```

Realização de transações com fragmentos

- Antes de chamar commit(), pode-se chamar addToBackStack() para adicionar a transação a uma pilha de retorno de transações de fragmentos.
- A pilha de retorno é gerenciada pela atividade e permite que o usuário retorne ao estado anterior do fragmento ao pressionar o botão Voltar.
- Desta forma, pode-se substituir um fragmento por outro e preservar o estado anterior da pilha de retorno.

Realização de transações com fragmentos

```
// Create new fragment and transaction
Fragment newFragment = new ExampleFragment();
FragmentTransaction transaction = getFragmentManager().beginTransaction();

// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack
transaction.replace(R.id.fragment_container, newFragment);
transaction.addToBackStack(null);

// Commit the transaction
transaction.commit();
```

Comunicação com a atividade

- Uma dada instância de um fragmento é diretamente vinculada à atividade que o contém,
- O fragmento pode acessar a instância Activity com getActivity() e realizar facilmente tarefas como encontrar uma exibição no layout da atividade
- Da mesma forma, a atividade pode chamar métodos no fragmento adquirindo uma referência para o Fragment no FragmentManager usando findFragmentById() ou findFragmentByTag().

```
View listView = getActivity().findViewById(R.id.list);
ExamplFrag fragment = (ExamplFrag)getFragmentManager().
  findFragmentById(R.id.example_fragment);
```

Processamento do ciclo de vida dos fragmentos

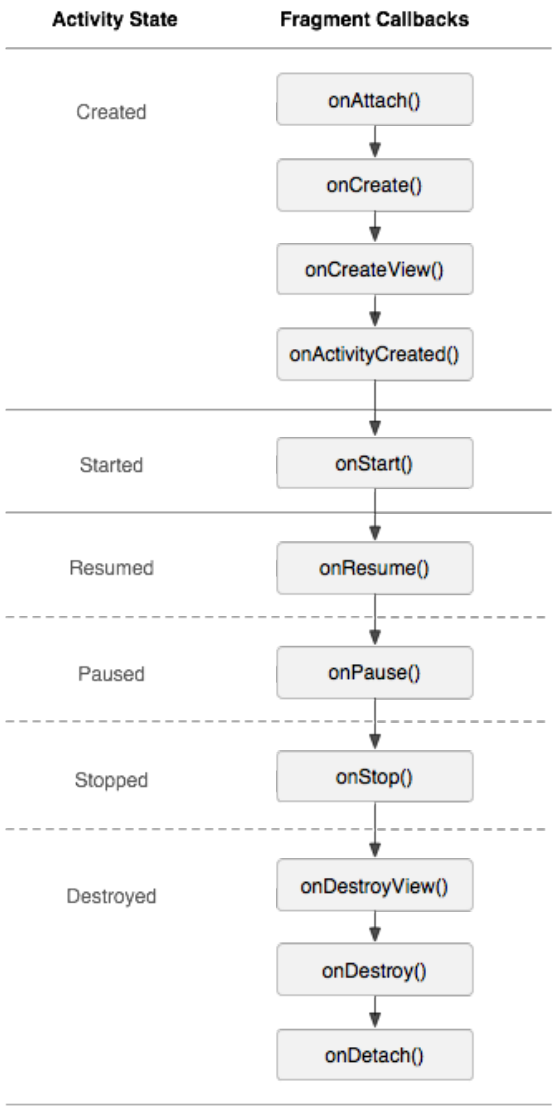
- Um fragmento pode existir em três estados:
  - **Resumed:** O fragmento é visível na atividade em execução.
  - **Paused:** Outra atividade está em primeiro plano, mas a atividade em que esse fragmento se encontra ainda está visível (a atividade de primeiro plano é parcialmente transparente ou não cobre a tela inteira).
  - **Stopped:** O fragmento não é visível. A atividade do host foi interrompida ou o fragmento foi removido da atividade mas adicionado à pilha de retorno. Um fragmento interrompido ainda está ativo (todas as informações do membro e de estado estão retidas no sistema).

Processamento do ciclo de vida dos fragmentos

- É possível reter o estado de um fragmento usando um Bundle se o processo da atividade for eliminado e for preciso restaurar o estado do fragmento.
- Pode-se salvar o estado durante o retorno de chamada de onSaveInstanceState() do fragmento e restaurá-lo durante onCreate(), onCreateView() ou onActivityCreated().
- A diferença mais significativa entre o ciclo de vida de uma atividade e de um fragmento é o armazenamento nas respectivas pilhas de retorno.
  - Por padrão, uma atividade é colocada de volta na pilha de retorno de atividades, gerenciada pelo sistema, quando interrompida (para que o usuário possa navegar a ela com o botão Back).

Android

Ciclo de Vida dos Fragmentos



Exemplo