CEFET-MG
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Aplicações Móveis

Aula 7 - Exibição de listas - ListView

Prof. Dr. Wendell Fioravante da Silva Diniz
3º Ano - Informática Integrado
Centro Federal de Educação Tecnológica de Minas Gerais
Unidade Varginha

Recapitulando...

- Listviews são usados para exibição de itens em lista
- Os dados são gerenciados pelos Adapters
- O SDK provê alguns Adapters prontos
- É possível personalizar um Adapter para lidar com tipos de dados diferentes
- O layout da lista pode ser construído no editor

Passagem de objetos entre atividades

- A classe Bundle tem métodos para passagem de dados de tipos primitivos
  - Bundle.putString()
  - Bundle.putInt()
  - Bundle.putFloat()...
- Mas e se precisarmos passar um tipo abstrato (criado pelo desenvolvedor)?
- Para isto, precisamos “ensinar” o tipo abstrato a salvar o seu estado (valores de seus atributos) em um arquivo e remontar o objeto a partir deste arquivo (serialização)
- A classe Parcelable fornece uma interface com essas funcionalidades

Passagem de objetos entre atividades

```java
public class Pessoa {

    private String nome;
    private String telefone;
    private int imagem;

    public Pessoa(String nome, String telefone, int idImagem) {
        this.nome = nome;
        this.telefone = telefone;
        this.imagem = idImagem;
    }

    public String getNome() { return nome;}

    public void setNome(String nome) { this.nome = nome;}

    public String getTelefone() { return telefone;}

    public void setTelefone(String telefone) {this.telefone = telefone;}

    public int getImagem() { return imagem;}

    public void setImagem(int id) { this.imagem = id;}

}
```

A classe Parcelable

```
1  public class MyParcelable implements Parcelable {
2      private int mData;
3
4      public int describeContents() {
5          return 0;
6      }
7
8      public void writeToParcel(Parcel out, int flags) {
9          out.writeInt(mData);
10     }
11
12     public static final Parcelable.Creator<MyParcelable> CREATOR
13             = new Parcelable.Creator<MyParcelable>() {
14         public MyParcelable createFromParcel(Parcel in) {
15             return new MyParcelable(in);
16         }
17
18         public MyParcelable[] newArray(int size) {
19             return new MyParcelable[size];
20         }
21     };
22
23     private MyParcelable(Parcel in) {
24         mData = in.readInt();
25     }
26 }
```

CEFET-MG

Adicionando o Parcelable ao tipo abstrato

```java
import android.os.Parcel;
import android.os.Parcelable;

public class Pessoa implements Parcelable {

    ...

    protected Pessoa(Parcel in) {
        nome = in.readString();
        telefone = in.readString();
        imagem = in.readInt();
    }

    @Override
    public int describeContents() {
        return 0;
    }
}
```

```java
@Override
public void writeToParcel(Parcel dest, int flags) {
    dest.writeString(nome);
    dest.writeString(telefone);
    dest.writeInt(imagem);
}

@SuppressWarnings("unused")
public static final Parcelable.Creator<Pessoa> CREATOR =
    new Parcelable.Creator<Pessoa>() {
    @Override
    public Pessoa createFromParcel(Parcel in) {
        return new Pessoa(in);
    }

    @Override
    public Pessoa[] newArray(int size) {
        return new Pessoa[size];
    }
};
```

Passando os dados

```java
public class MainActivity extends AppCompatActivity {

    ArrayList<Pessoa> lista = new ArrayList<>();

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

        lista.add(new Pessoa("Maria de Oliveira", "99323-1234", R.mipmap.ic_launcher_round));
        lista.add(new Pessoa("Pedro da Silva", "3690-1234", R.mipmap.ic_launcher_round));
        lista.add(new Pessoa("Joao de Souza", "3690-4321", R.mipmap.ic_launcher_round));
        Log.d("Lista: ", Integer.toString(lista.size()));
    }

    public void mostrarLista(View v) {
        Intent it = new Intent(this, mostraListaDinamica.class);
        Bundle bundle = new Bundle();
        bundle.putParcelableArrayList("contatos", lista);
        it.putExtras(bundle);
        startActivity(it);
    }
}
```

Recebendo os dados

```
1  public class mostraListaDinamica extends AppCompatActivity {
2
3      @Override
4      protected void onCreate(Bundle savedInstanceState) {
5          super.onCreate(savedInstanceState);
6          setContentView(R.layout.activity_mostra_lista_dinamica);
7
8          ListView listview = findViewById(R.id.listView);
9
10         Intent intent = getIntent();
11         Bundle bundle = intent.getExtras();
12
13         try {
14             ArrayList<Pessoa> lista = bundle.getParcelableArrayList(
15                     "contatos");
16             PessoaAdapter adapter = new PessoaAdapter(
17                     mostraListaDinamica.this, lista);
18             listview.setAdapter(adapter);
19         }
20         catch (Exception e){
21             Log.d(e.getClass().toString(), e.getMessage());
22         }
23
24     }
25 }
```

CEFET-MG

Tratando exceções

- Exceções são eventos que interrompem o fluxo de execução de um programa
- Servem para sinalizar e tratar erros em tempo de execução
- Quando ocorre uma exceção, um objeto do tipo Exception é criado e passado para o sistema tratador
- Na Exception, são coletadas informações sobre a natureza do erro

Exceções em Java

- O sistema de exceções em Java pode ser resumido desta forma:
  a. Quando um erro acontece dentro de um método, a Exception é gerada com as informações do estado de execução do programa e passado para o sistema de tratamento
  b. O sistema então busca na pilha de chamadas, um método que possa tratar a exceção que foi gerada
  c. Se um método com o tratamento para a exceção é encontrado, o objeto Exception é passado e o tratamento é executado
  d. Caso o sistema navegue por toda a pilha de chamadas sem encontrar um método tratador, o programa é encerrado

Exceções em Java

```
1  try {
2      // Bloco de código que pode
3      // lançar a exceção
4
5  } catch (Exception e) {
6      // Tratamento de exceção genérico
7
8  } catch (NullPointerException e) {
9      // Tratamento de exceção específico
10
11 }
```

Exceções em Java

- Boas práticas para um bom sistema de exceções:
  a. Não ignorar exceções: um catch vazio invalida o propósito do sistema de exceções
  b. Use subtipos de Exception para exceções tratáveis: permita que o programa se recupere usando os subtipos efetivamente
  c. Erros representam condições irrecuperáveis e não devem ser tratados: são causados por falhas no ambiente de execução e não devem ser tratadas (exemplo: falta de memória)
  d. Uma exceção deve definir precisamente a natureza do problema: evite o uso de exceções genéricas, use as exceções definidas na linguagem ou crie suas próprias exceções para precisar a natureza do problema
  e. Documente as exceções lançadas pelo método: outros desenvolvedores que usarão seu código devem estar cientes das exceções que um método pode lançar
- Ao usar o SDK, consulte a documentação para descobrir as exceções que um determinado método pode lançar

Exceções em Java

```
1  public class mostraListaDinamica extends AppCompatActivity {
2
3      @Override
4      protected void onCreate(Bundle savedInstanceState) {
5          super.onCreate(savedInstanceState);
6          setContentView(R.layout.activity_mostra_lista_dinamica);
7
8          ListView listview = findViewById(R.id.listView);
9
10         Intent intent = getIntent();
           Bundle bundle = intent.getExtras();

           try {
               ArrayList<Pessoa> lista = bundle.getParcelableArrayList(
                       "contatos");
               PessoaAdapter adapter = new PessoaAdapter(
                       mostraListaDinamica.this, lista);
               listview.setAdapter(adapter);
           }
           catch (Exception e){
               Log.d(e.getClass().toString(), e.getMessage());
           }

24     }
25 }
```

A documentação diz que o método *getParcelableArrayList* pode lançar uma exceção

Os métodos do objeto Exception retornam informações sobre o erro

CEFET-MG

O sistema de build Gradle

- Gradle é um sistema de builds flexível que permite gerenciar dependências de forma simples
- Ele automatiza tarefas comuns para o build (compilação) do projeto
- Entre outras tarefas, ele cuida da sincronização e da geração da classe R (gerenciador de recursos)

O arquivo app/build.gradle

- Neste arquivo, ficam as configurações de dependência do seu projeto
- É dividido em seções, de acordo com a configuração a ser controlada
- Pode-se definir também diferentes tipos de build

O arquivo app/build.gradle

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "com.example.wendell.listadinmica"
        minSdkVersion 19
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
}
```

O arquivo app/build.gradle

Identificação do tipo de build

```
1  apply plugin: 'com.android.application'
2
3  android {
4      compileSdkVersion 26
5      defaultConfig {
6          applicationId "com.example.wendell.listadinmica"
7          minSdkVersion 19
8          targetSdkVersion 26
9          versionCode 1
10         versionName "1.0"
11         testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
12     }
13     buildTypes {
14         release {
15             minifyEnabled false
16             proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
17         }
18     }
19 }
20
21 dependencies {
22     implementation fileTree(dir: 'libs', include: ['*.jar'])
23     implementation 'com.android.support:appcompat-v7:26.1.0'
24     implementation 'com.android.support.constraint:constraint-layout:1.0.2'
25     testImplementation 'junit:junit:4.12'
26     androidTestImplementation 'com.android.support.test:runner:1.0.1'
27     androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
28 }
```

O arquivo app/build.gradle

Configurações de build do projeto

```
1  apply plugin: 'com.android.application'
2
3  android {
4      compileSdkVersion 26
5      defaultConfig {
6          applicationId "com.example.wendell.listadinmica"
7          minSdkVersion 19
8          targetSdkVersion 26
9          versionCode 1
10         versionName "1.0"
11         testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
12     }
13     buildTypes {
14         release {
15             minifyEnabled false
16             proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
17         }
18     }
19 }
20
21 dependencies {
22     implementation fileTree(dir: 'libs', include: ['*.jar'])
23     implementation 'com.android.support:appcompat-v7:26.1.0'
24     implementation 'com.android.support.constraint:constraint-layout:1.0.2'
25     testImplementation 'junit:junit:4.12'
26     androidTestImplementation 'com.android.support.test:runner:1.0.1'
27     androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
28 }
```

O arquivo app/build.gradle

Configuração de dependências externas

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "com.example.wendell.listadinmica"
        minSdkVersion 19
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJ    Runner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt  , 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
}
```

Adicionando dependências

- Muitas bibliotecas de terceiros estão disponíveis para adicionar funcionalidades
- Para usá-las, é necessário incluir suas informações no arquivo build.gradle
- Possivelmente, também deve ser adicionado o repositório público onde consegui-las
- O repositório padrão do Android SDK é Maven JCenter
- Se não é indicado nenhum repositório, o padrão é assumido

Adicionando dependências

```
1  // Top-level build file where you can add configuration options common to all sub-projects/modules.
2
3  buildscript {
4
5      repositories {
6          google()
7          jcenter()
8      }
9      dependencies {
10         classpath 'com.android.tools.build:gradle:3.0.1'
11
12
13         // NOTE: Do not place your application dependencies here; they belong
14         // in the individual module build.gradle files
15     }
16 }
17
18 allprojects {
19     repositories {
20         google()
21         jcenter()
22     }
23 }
24
25 task clean(type: Delete) {
26     delete rootProject.buildDir
27 }
```

Adicionando dependências

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "com.example.wendell.qrcodescanner"
        minSdkVersion 19
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.journeyapps:zxing-android-embedded:3.4.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
}
```

Leitura complementar recomendada

- Capítulo 38 do livro **Google Android** de Ricardo Lecheta

CEFET-MG