FACULDADE IBRATEC

Android: Persistência de dados

Prof. Rodrigo Barbosa rodrigo.bezerra@unibratec.edu.br

Persistência de dados

- O Android fornece várias opções para salvar os dados dos aplicativos
- A escolha da solução vai depender da necessidade do aplicativo (se os dados devem ser privados de sua aplicação ou acessíveis a outras aplicações)

Persistência de dados

- SharedPreferences: Armazenar dados particulares primitivos em pares chave-valor
- Internal Storage: Armazenar dados privados na memória do dispositivo (com persistência de objetos)
- External Storage: Armazenar dados públicos sobre o armazenamento externo compartilhado
- SQLite Databases: Armazenar dados estruturados em um banco de dados privados

SharedPreferences

- A classe SharedPreferences permite salvar e recuperar pares de chave/valor de tipos de dados primitivos (boolean, float, int, long, string)

- Para obter um objeto SharedPreferences utilize um dos dois métodos

  - getSharedPreferences: Utilize se precisar de vários arquivos de preferências
  - getPreferences: Utilize se precisar de um único arquivo de preferência

SharedPreferences

- Para escrever valores
  - Usar o método edit() para obter uma SharadPreferences.Editor
  - Adicionar valores com métodos tais como putBoolean() e putString()
  - Persistir os novos valores com o método commit()
- Para ler os valores utilize os métodos como getBoolean() e getString()

FACULDADE IBRATEC®

Exercício 18

salvar e recuperar preferências

Internal Storage

- Por padrão, os arquivos salvos para o armazenamento interno são privados de sua aplicação, fazendo com que outros aplicativos não possam acessá-los

- Quando o usuário desinstala o aplicativo os arquivos são removidos

Internal Storage

- Para criar e gravar um arquivo privado para o armazenamento interno devemos
  - Usar openFileOutput() com o nome do arquivo e o modo de funcionamento (MODE_PRIVATE), teremos um FileOutputStream como retorno
  - Escrever no arquivo usando o write()
  - Fechar o fluxo com close()

Internal Storage

```
String FILENAME = "file";
String text = "my text";

FileOutputStream fos = null;
try {
    fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
    fos.write(text.getBytes());
    fos.close();
} catch (IOException e) {
    e.printStackTrace();
}
```

Internal Storage

- Podemos gravar qualquer objeto em disco, mas para isso ele deve implementar a interface Serializable do pacote java.io

- Isso permite construir um objeto padrão de configuração, controlando melhor os dados persistidos

Internal Storage

```java
public class MyObject implements Serializable{
    private int mCode;
    private String mDescription;

    public MyObject(int code, String description){
        mCode = code;
        mDescription = description;
    }

    public int getCode(){
        return mCode;
    }

    public void setCode(int code){
        mCode = code;
    }

    public String getDescription(){
        return mDescription;
    }

    public void setDescription(String description){
        mDescription = description;
    }
}
```

```java
String FILENAME = "file";
File file = getFileStreamPath(FILENAME);

//PARA GRAVAR
try{
    FileOutputStream fos = new FileOutputStream(file);
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(OBJECT);
    oos.close();
    fos.close();
} catch(IOException e){
    e.printStackTrace();
}
```

```java
String FILENAME = "file";
File file = getFileStreamPath(FILENAME);

//PARA LER
try{
    FileInputStream fis = new FileInputStream(file);
    ObjectInputStream ois = new ObjectInputStream(fis);
    MyObject result = (MyObject) ois.readObject();
    fis.close();
    ois.close();
} catch(IOException e){
    e.printStackTrace();
}
```

External Storage

- Dispositivos Android tem uma “memória externa” compartilhada que pode ser usada para salvar arquivos
- Pode ser uma mídia de armazenamento removível (cartão SD) ou memória interna (não removível)
- Os arquivos são de leitura para todos e podem ser modificados pelo usuário
- Antes de usar o armazenamento externo, devemos sempre usar o Environment.getExternalStorageState(), para verificar se a mídia está disponível

External Storage

```
boolean mExternalStorageAvailable = false;
boolean mExternalStorageWriteable = false;

if(Enviroment.MEDIA_MOUNTED.equals(state)){
    //Podemos ler e escrever os meios de comunicação
    mExternalStorageAvailable = mExternalStorageWriteable = true;
} else if(Enviroment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
    //Só podemos ler
    mExternalStorageAvailable = true;
    mExternalStorageWriteable = false;
} else {
    //Pode ser um de muitos outros estados
    //Só precisamos saber que não pode ler e nem escrever
    mExternalStorageAvailable = mExternalStorageWriteable = false;
}
```

External Storage

```
File dir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
File file = new File(dir, "teste.obj");

FileOutputStream fos = null;
try{
    fos = new FileOutputStream(file);
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(OBJECT);
    oos.close();
    fos.close();
}catch(IOException e){
    e.printStackTrace();
}
```

SQLite Databases

- O Android utiliza o banco de dados SQLite que é open-source e muito utilizado em aplicações populares (O SQLite também é utilizado pelo Firefox e Iphone)
- Um banco de dados criado para uma aplicação só é acessível para a mesma
- O banco é armazenado no diretório data/data/nome_do_pacote/databases/nome_do_banco
- O banco pode ser gerenciado via código e via adb (utilizando a ferramenta sqlite3)

SQLite Databases

- O recomendado para criar um banco de dados SQLite novo é criar uma subclasse de SQLiteOpenHelper e sobrescrever o método OnCreate()

SQLite Databases

```
public class Database extends SQLiteOpenHelper{
    public Database(Context context, String name, int version){
        super(context, name, version);
    }

    @Override
    public void onCreate(SQLiteDatabase sqld){
        sqld.execSQL("CREATE TABLE usuarios_tbl ("
                + "id_usuarios INTEGER PRIMARY KEY autoincrement,"
                + "usuario varchar(45) NOT NULL,"
                + "senha varchar(45) NOT NULL,"
                + "nome_completo varchar(45) NOT NULL"
                + ");");
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqld, int i, int il){
        //TODO
    }
}
```

SQLite Databases

- Para utilizar devemos instanciar passando o nome e a versão (inicia em 1), se a versão for alterada o método onUpgrade é chamado

- Depois de instanciado temos acesso ao banco através do método getWritableDatabase(), como ele podemos fazer insert, update, select e etc

SQLite Databases

```
//Construção do banco
Database banco = new Database(this, "blablabla", 1);

//Insert
ContentValues contentValues = new ContentValues();
contentValues.put("usuario", "rodrigo");
contentValues.put("senha", "123456");
contentValues.put("nome_completo", "Rodrigo Barbosa de Lima Bezerra");
banco.getWritableDatabase().insert("usuarios_tbl", null, contentValues);

//Select nas colunas usuario, nome_completo
Cursor cursor = banco.getWritableDatabase().query("usuarios_tbl", new String[]{"usuario", "nome_completo"}, null, null, null, null, null);
while(cursor.moveToNext()){
    Log.i("rblb", "usuario: " + cursor.getString(0));
    Log.i("rblb", "nome completo: " + cursor1.getString(1));
}

//Lembre-se de sempre fechar o cursor
cursor.close();
```

FACULDADE IBRATEC®

Exercício 19

Fazer App de cadastro