



Programação para Android

Aula 07: Persistência de dados - Shared Preferences, Internal e External Storage

Persistência de dados

- ▶ Na maioria das aplicações precisamos ter algum tipo de persistência de dados.
- ▶ Para guardar informações de forma persistente no Android podemos usar os seguintes recursos:
 - ▶ *Shared Preferences*: Armazena dados primitivos privados em pares chave-valor.
 - ▶ *Internal Storage*: Armazena dados privados na memória do dispositivo.
 - ▶ *External Storage*: Armazena dados públicos no cartão de memória.
 - ▶ *SQLite Databases*: Armazena dados estruturados num banco de dados privado.
 - ▶ *Network Connection*: Armazena dados na web no seu servidor de rede.

Parte 01: SharedPreferences

Persistência de dados - Shared preferences

- ▶ A classe `SharedPreferences` é utilizada para armazenar informações, na qual é possível salvar entradas do tipo *chave-valor*, onde se associa um “nome” a uma determinada informação para que depois se possa recuperá-la através deste nome.
- ▶ Podemos usar `SharedPreferences` para salvar qualquer tipo primitivo: `boolean`, `floats`, `ints`, `longs`, e `strings`.
- ▶ O Android salva tudo em um arquivo `XML` dentro da estrutura interna “de disco” em cada aplicação. O Android também oferece funções para a tarefa de salvar e depois buscar novamente o que foi salvo.
- ▶ Essa opção é indicada para poucas informações e que sejam simples, como números, *strings* e valores *booleanos*. Por isso esse mecanismo é geralmente utilizado para armazenar as preferências que o usuário definiu na aplicação.

Persistência de dados - Shared preferences

- ▶ Para registrar um valor no SharedPreferences primeiro criamos (ou abrimos se já existir) o *SharedPreferences* informando o nome do arquivo e o modo de acesso que ele deve ter.
- ▶ Depois criamos um *Editor*, que nada mais é que uma classe auxiliar para escrever no arquivo, e salvamos o que foi passado, informando uma chave para cada informação.
- ▶ Por último realizamos o *commit*, para efetivamente escrever tudo no arquivo.

Persistência de dados - Shared preferences

- ▶ Código fonte para registrar/escrever uma chave-valor no Shared Preferences:

```
// Cria ou abre.  
    SharedPreferences prefs = getSharedPreferences("preferencia",  
Context.MODE_PRIVATE);  
  
// Precisamos utilizar um editor para alterar Shared Preferences.  
    Editor ed = prefs.edit();  
  
// salvando informações de acordo com o tipo  
    ed.putString("USUARIO", "Regilan Silva");  
    ed.putString("IDADE", 30);  
  
// Grava efetivamente as alterações.  
    ed.commit();
```

Persistência de dados - Shared preferences

- ▶ Mais simples que escrever no *Shared Preferences* é ler as informações dele.
- ▶ Basta acessá-lo da mesma forma e recuperar cada informação pela sua chave identificadora (o seu “nome”), informando após o que deve retornar caso nada seja encontrado para aquela chave.

```
// Acesso às Shared Preferences usando o nome definido.  
SharedPreferences prefs = getSharedPreferences("preferencia",  
Context.MODE_PRIVATE);
```

```
// Acesso às informações de acordo com o tipo.  
String texto = prefs.getString("TEXTO", "não encontrado");  
String idade = prefs.getString("IDADE", "não encontrado");
```

Exemplo 01 - Persistência de dados com Shared Preferences



The screenshot shows an Android application interface with a black header bar containing an Android robot icon and the text 'Aula07'. Below the header, there are two text input fields. The first field is labeled 'Nome:' and contains the text 'Regilan Silva'. The second field is labeled 'Idade:' and contains the number '30'. Below these fields are two gray buttons: 'Gravar dados' (Save data) and 'Exibir dados' (Display data). At the bottom of the screen, the text 'Regilan Silva' and '30' is displayed, indicating that the data has been successfully saved and retrieved.

Aula07

Nome:
Regilan Silva

Idade:
30

Gravar dados

Exibir dados

Regilan Silva
30

Parte 02: Internal e External Storage

Storage

- ▶ A opção do *Storage* nada mais é que um espaço “em disco” que cada aplicação tem onde é possível salvar arquivos.
- ▶ Existe a opção do *Internal Storage* (espaço na estrutura de arquivos interna da aplicação, que é o mesmo onde fica(m) o(s) arquivo(s) de *Shared Preferences*) e do *External Storage*, que geralmente é um espaço no *SDCard* - podendo ser público (pastas de música ou fotos por exemplo) ou da aplicação - e nem sempre estará disponível (se o *SDCard* for removido, por exemplo).
- ▶ Existem várias maneiras (classes) disponíveis de manipulação de arquivos, utilizaremos uma maneira mais simples de acessar dados de arquivo.

Internal Storage - Escrita de dados

- ▶ Primeiro criamos um arquivo no diretório do *Internal Storage* (retornado pelo método `getFilesDir()`) e depois criamos uma instância da classe auxiliar de escrita no arquivo.
- ▶ Nesta classe auxiliar usamos o modo *append* (escreve a partir do fim do arquivo, sem apagar ou sobrescrever o que já existia antes). Escrevemos uma informação em cada linha (escrevendo a “quebra de linha”) para facilitar a leitura posteriormente, e efetivamos a escrita das informações no arquivo.
- ▶ O acesso a arquivos em disco pode gerar erros em vários pontos, por isso recomendamos o uso do tratamento de exceções (`try...catch`). O Netbeans automaticamente pedirá para incluir o código dentro de uma estrutura de tratamento de exceções.

Internal Storage - Escrita de dados

► Código fonte para escrita de dados em arquivo:

```
// Cria o arquivo onde serão salvas as informações.
File arquivo = new File(getFilesDir().getPath()+ "/dados.txt");
// Cria a classe auxiliar para manipular arquivos.
FileWriter escrever;
escrever = new FileWriter(arquivo, true);

escrever.append(nome); // Registra um valor no arquivo.
escrever.append("\n"); // Quebra de linha.
escrever.append(idade);
escrever.append("\n"); // Quebra de linha.

// Escreve no arquivo.
escrever.flush();
// Fecha o arquivo para escrita de dados.
escrever.close();
```

Internal Storage - Leitura de dados

- ▶ Assim como ocorreu na escrita de dados, passamos o caminho onde criamos o arquivo e utilizamos uma classe auxiliar (dessa vez de leitura de arquivos), além de um buffer que vai nos permitir fazer a leitura linha a linha (por isso salvamos separando por linhas) sem nos preocuparmos com a quantidade que deve ser lida por vez.
- ▶ Utilizamos então um “laço” para ler o arquivo até o seu fim (quando o método *readLine()* retornar *null*).
- ▶ OBS: Cada aplicativo possui uma pasta com o seu nome no caminho **/data/data/**. Neste local encontramos todos os arquivos que foram criadas para a aplicação.

Internal Storage - Leitura de dados

- Código fonte para leitura de dados em arquivo:

```
FileReader ler;  
BufferedReader bufferDados;  
String dados="";  
ler = new FileReader(getFilesDir().getPath()+"/dados.txt");  
  
bufferDados = new BufferedReader(ler);  
  
String linha;  
  
while ((linha = bufferDados.readLine()) != null) {  
    dados += linha;  
}  
  
bufferDados.close();
```

Exemplo 02 - Persistência de dados com Internal Storage



A screenshot of an Android application interface. At the top, there is a black header bar with a green Android robot icon and the text "Aula07". Below the header, the form contains two labels: "Nome:" and "Idade:". The "Nome:" label is followed by a text input field containing the text "Regilan Silva". The "Idade:" label is followed by a text input field containing the number "30". Below the input fields, there are two gray buttons with black text: "Gravar dados" and "Exibir dados".

External Storage

- ▶ Para utilizar o *External Storage*, precisaríamos apenas testar antes se ele está disponível e indicar o caminho correto na criação do arquivo. Para escrever nesse local ainda precisamos adicionar uma permissão no *Manifest* da aplicação.
- ▶ Os arquivos salvos na memória externa podem ser lidos por qualquer outro dispositivo ou aplicação e podem ser modificados pelo usuário quando ele ativar a transferência USB ao conectar o dispositivo com um computador.
- ▶ Arquivos armazenados na memória externa pode desaparecer se o usuário montar a mídia em um computador ou remover o cartão, e não existe nenhuma medida de segurança sobre os arquivos salvos nesse tipo de memória. Todas as aplicações podem ler e escrever arquivos localizados em memórias externas e o usuário pode remove-los.
- ▶ RECOMENDAMOS USAR SHARED PREFERENCES OU INTERNAL STORAGE

Na próxima aula...

- Persistência de dados: banco de dados SQL Lite

