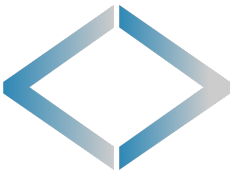

SharedPreferences

Conceito

É uma funcionalidade para armazenamento de dados relativamente pequenos e que precisam estar disponíveis facilmente. Não deve ser usado para fazer o gerenciamento de dados, como por exemplo um gerenciamento de clientes. Para esses casos, onde existe uma estrutura e relações entre os dados, deve ser usado um banco de dados.

É uma estrutura de dados no formato chave-valor e que pode ser criada com configuração de privacidade. Abaixo estão as opções disponíveis com uma breve explicação:

- **MODE_PRIVATE**: Somente a aplicação tem permissão de leitura e escrita nos dados armazenados ou aplicações que compartilham o mesmo user ID.

Como iniciar SharedPreferences

É possível criar um arquivo SharedPreferences da seguinte maneira:

```
SharedPreferences shared =
    context.getSHaredPreferences("KEY", Context.MODE_PRIVATE);
```

É importante destacar que para obter um arquivo SharedPreferences é necessário a utilização de um *Context,* ele não é instanciado a partir da classe SharedPreferences. Vale destacar também que o primeiro parâmetro (*KEY*) é a chave do arquivo criado e somente com ela o mesmo arquivo pode ser recuperado. Caso a chave usada seja diferente desse valor, basta usá-la na implementação para acessar o arquivo criado.

Escrever no arquivo SharedPreferences

Para escrever um valor, basta obter o arquivo de SharedPreferences através da chave *KEY* e através do método edit() informar qual a chave e o valor desejado para ser armazenado.

A partir desse momento, o valor salvo estará disponível através da chave que foi previamente salva.

```
SharedPreferences shared =
    context.getSHaredPreferences("KEY", Context.MODE_PRIVATE);
shared.edit().putString("KEY", "Value").apply();
```

Ler do arquivo SharedPreferences

Para obter o valor salvo associado a uma chave basta obter o arquivo de SharedPreferences através da chave *KEY* e usar o método *getString* para obter o valor salvo através da chave.

Muito importante observar que existem outros métodos para salvar outros tipos de variáveis, por exemplo putByte, putInteger, etc. Dessa maneira, caso uma variável de tipo diferente tenha sido salva, basta usar o método correspondente para obter o valor salvo, por exemplo getByte, getInteger, etc.

Além disso, o segundo parâmetro da obtenção do valor salvo, representa um valor padrão caso essa chave não tenha nenhum valor associado. Ou seja, caso o valor seja nulo é possível definir um valor padrão de retorno para evitar erros de NullPointerException.

```
SharedPreferences shared =
    context.getSHaredPreferences("KEY", Context.MODE_PRIVATE);
shared.getString("KEY", "");
```