07

Salvando versão com o shared preferences

Transcrição

Agora que conseguimos fazer a nossa refatoração, precisamos pegar os dados do servidor que representa a versão dos nossos dados e persista na aplicação.

Mas qual é a informação que temos no servidor que representa essa versão dos nossos dados? Se olharmos o *Log* do Android Monitor, podemos ver que temos uma propriedade chamada "momentoDaUltimaModificacao". É justamente esse campo que vamos salvar na aplicação.

Como podemos pegar essa informação? Nos cursos anteriores, nós vimos que o `AlunoSync` é um **DTO** que representa todo o corpo da resposta que vem do servidor.

No método `onResponse()` da classe `AlunoSincronizador` chamamos a o método `alunoSync.getMomentoDaUltimaVerificação();` e guardamos em uma variável.

```
// ...

@Override
public void onResponse(Call<AlunoSync> call, Response<AlunoSync> response){
    AlunoSync alunoSync = response.body();
    String versão = alunoSync.getMomentoDaUltimaVerificação();

    AlunoDAO dao = new AlunoDAO(context);
    dao.sincroniza(alunoSync.getAlunos());
    dao.close();
    bus.post(new AtualizaListaAlunoEvent());
    context.getSwipe().setRefreshing(false);
}

// ...
```

Já temos a nossa versão. Agora poderemos persistir a informação, mas qual seria a forma mais adequada? Estamos persistindo os nossos dados com o **SQLite**, mas não faz sentido criarmos uma tabela para guardar apenas uma informação, SQLite é mais para entidades.

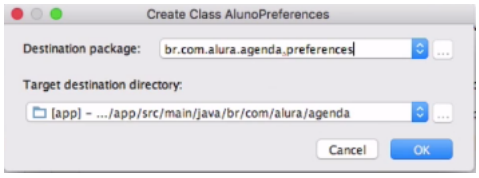
Felizmente o Android tem diversas opções de armazenamento, as ***Storage Options***. E a opção chamada de ***Shared Preferences*** se aplica a nossa situação de armazenar um dado primitivo. O *Shared Preferences* é um framework feito para armazenar os tipos primitivos.

O *Shared Preferences* não é o tipo de responsabilidade da classe `AlunoSincronizador`, já que ela só tem responsabilidade de mandar e buscar informações no servidor, ou fazer alguma comunicação externa. Então vamos criar uma classe resposavel pelo *Shared Preferences*.

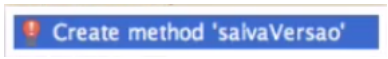
Vamos criar então uma classe chamada `AlunoPreferences`, onde a sua função é guardar informações de *Shared Preferences* para aluno. Então vamos instanciar a classe e chamar a ação que queremos com `new AlunoPreferences().salvaVersao(versao);`.

```
// ...

@Override
public void onResponse(Call<AlunoSync> call, Response<AlunoSync> response){
    AlunoSync alunoSync = response.body();
    String versão = alunoSync.getMomentoDaUltimaVerificação();

    new AlunoPreferences().salvaVersao(versao);

    AlunoDAO dao = new AlunoDAO(context);
    dao.sincroniza(alunoSync.getAlunos());
    dao.close();
    bus.post(new AtualizaListaAlunoEvent());
    context.getSwipe().setRefreshing(false);
}

// ...
```

Como ainda não criamos a classe, podemos usar os recursos da IDE para criá-la. Colocamos o *cursor* do editor em cima da instancia `AlunoPreferences` e usamos o atalho "Alt + Enter". Na janela aberta colocaremos o pacote como `br.com.alura.agenda.preferences` e apertamos "OK".



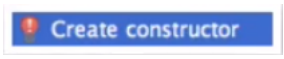
Criamos a nossa nova classe, porém sem o método `salvaVersao()`. Voltamos a classe `AlunoSincronizador` e colocamos o *cursor* do editor em cima do método e usamos novamente o "Alt + Enter", selecionamos a opção `Create method 'salvaVersao'` e teremos o método criado na classe `AlunoPreferences`.



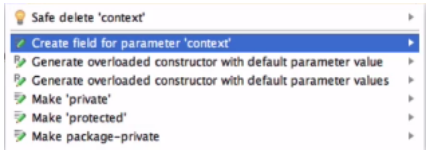
Para acessarmos o *Shared Preferences* na nova classe, precisamo de uma referência do `Context`, então vamos passar no construtor a variável `context`.

```
// ...

@Override
public void onResponse(Call<AlunoSync> call, Response<AlunoSync> response){
    AlunoSync alunoSync = response.body();
    String versão = alunoSync.getMomentoDaUltimaVerificação();

    new AlunoPreferences(context).salvaVersao(versao);

    AlunoDAO dao = new AlunoDAO(context);
    dao.sincroniza(alunoSync.getAlunos());
    dao.close();
    bus.post(new AtualizaListaAlunoEvent());
    context.getSwipe().setRefreshing(false);
}

// ...
```

Como não temos nenhum construtor criado na classe `AlunoPreferences`, então basta colocar o *cursor* do editor em cima da variável `context` que estamos passando no construtor e usar o "Alt + Enter".



Dentro da classe `AlunoPreferences` colocamos o *cursor* em cima do parâmetro `context` do construtor e usamos mais uma vez o atalho "Alt + Enter", selecionando a opção ***Create field for parameter 'context'***.



Pronto, já temos a classe, o construtor, o atributo e o método criados. Nossa classe no momento está assim:

```
package br.com.alura.agenda.preferences

import android.content.Context;

public class AlunoPreferences{

    private Context context;

    public AlunoPreferences(Context context){
        this.context = context;
    }

    public void salvaVersao(String versao){

    }
}
```

Dentro do método `salvaVersao()`, vamos chamar o *Shared Preferences* usando o `context.getSharedPreferences();`. Mas não é tão simples assim, precisamos ajustar alguns detalhes.

Precisamos passar como argumento uma *string* que vai ser o nome do conjunto do *Shared Preferences* que queremos acessar, no caso como é apenas a classe `AlunoPreferences` que vai acessar esse conjunto, então passaremos como chave `"br.com.alura.agenda.preferences.AlunoPreferences"`.

Qual o motivo de passar o contexto do pacote e o nome da classe como chave? Caso alguma outra classe ou biblioteca que faz uso do *Shared Preferences*, isso evita que elas acessem o nosso conjunto, já que ele vai estar usando outro tipo de chave.

Agora precisamos colocar o módulo, que também é acessado por meio do `context` também. Então vamos passar como segundo argumento o `context.MODE_PRIVATE`. Escolhemos privado para apenas a nossa aplicação tenha acesso.

Nossa classe no momento:

```
package br.com.alura.agenda.preferences

import android.content.Context;
```

```
public class AlunoPreferences{

    private Context context;

    public AlunoPreferences(Context context){
        this.context = context;
    }

    public void salvaVersao(String versao){
        context.getSharedPreferences("br.com.alura.agenda.preferences.AlunoPreferences", contex
    }
}
```

Repare que a nossa string é grande e prejudica na leitura do método. Como ela vai ser um valor constante, vamos extraí-la com o atalho "Ctrl + Alt + C". A IDE cria uma constante `public static final` , e para melhorar a legibilidade, vamos alterar o nome para `ALUNO_PREFERENCES` . Como não vamos acessá-la de outras classes, vamos mudar o modificador de acesso para `private` .

```
package br.com.alura.agenda.preferences

import android.content.Context;

public class AlunoPreferences{

    private static final String ALUNO_PREFERENCES = "br.com.alura.agenda.preferences.AlunoPrefe
    private Context context;

    public AlunoPreferences(Context context){
        this.context = context;
    }

    public void salvaVersao(String versao){
        context.getSharedPreferences(ALUNO_PREFERENCES, context.MODE_PRIVATE);
    }
}
```

Vamos associar o objeto retornado pelo método `getSharedPreferences()` em uma variável do tipo `SharedPreferences` chamada de `preferences` .

```
// ...

public void salvaVersao(String versao){
    SharedPreferences preferences = context.getSharedPreferences(ALUNO_PREFERENCES, context.MODE
}

// ...
```

E a partir do variável `preferences` chamamos o método `preferences.edit();` . Esse método retorna um objeto do tipo `SharedPreferences.Editor` que guardaremos em um variável chamada `editor` . Esse objeto retornado é o próprio editor do *Shared Preferences*.

```
// ...

public void salvaVersao(String versao){
    SharedPreferences preferences = context.getSharedPreferences(ALUNO_PREFERENCES, context.MODI
    SharedPreferences.Editor editor = preferences.edit();
}

// ...
```

Com o editor em mãos, podemos enviar para o *Shared Preferences* a informação que queremos. Existe diversas opções para armazenar tipos primitivos, mas como queremos armazenar uma *string*, então vamos utilizar o `editor.putString()` passando dois argumentos, uma chave e um valor. Como chave, utilizaremos a *string* `"versao_do_dado"` e o valor é a própria variável `versao`.

```
// ...

public void salvaVersao(String versao){
    SharedPreferences preferences = context.getSharedPreferences(ALUNO_PREFERENCES, context.MODI
    SharedPreferences.Editor editor = preferences.edit();
    editor.putString("versão_do_dado", versao);
}

// ...
```

Mais uma vez, vamos extrair a *string* `"versao_do_dado"` para uma constante usando o atalho "Ctrl + Alt + C", e colocar a nova constante como `private`.

Ainda não é o suficiente para persistir a informação, precisamos de mais um detalhe que é chamar o método `editor.commit();`.

```
package br.com.alura.agenda.preferences

import android.content.Context;

public class AlunoPreferences{

    private static final String ALUNO_PREFERENCES = "br.com.alura.agenda.preferences.AlunoPrefe
    private static final VERSAO_DO_DADO = "versão_do_dado";
    private Context context;

    public AlunoPreferences(Context context){
        this.context = context;
    }

    public void salvaVersao(String versao){
        SharedPreferences preferences = context.getSharedPreferences(ALUNO_PREFERENCES, context
        SharedPreferences.Editor editor = preferences.edit();
        editor.putString("versão_do_dado", versao);
        editor.commit();
    }
}
```

Agora conseguimos persistir nossas informações. Mas em nenhum momento de nossa aplicação, estamos conseguindo ter certeza de que os dados estão realmente sendo salvos.

Vamos verificar qual é o valor que está sendo salvo. Na classe `AlunoSincronizador` dentro do método `onResponse()` vamos colocar um `Log.i()` . Antes precisamos isolar a referência do `AlunoPreferences` em uma variável.

```
// ...

@Override
public void onResponse(Call<AlunoSync> call, Response<AlunoSync> response){
    AlunoSync alunoSync = response.body();
    String versão = alunoSync.getMomentoDaUltimaVerificação();

    AlunoPreferences preferences = new AlunoPreferences(context);
    preferences.salvaVersao(versao);

    AlunoDAO dao = new AlunoDAO(context);
    dao.sincroniza(alunoSync.getAlunos());
    dao.close();
    bus.post(new AtualizaListaAlunoEvent());
}

// ...
```

Agora usaremos o `Log.i("versao", preferences.getVersao());` .

```
// ...

@Override
public void onResponse(Call<AlunoSync> call, Response<AlunoSync> response){
    AlunoSync alunoSync = response.body();
    String versão = alunoSync.getMomentoDaUltimaVerificação();

    AlunoPreferences preferences = new AlunoPreferences(context);
    preferences.salvaVersao(versao);

    AlunoDAO dao = new AlunoDAO(context);
    dao.sincroniza(alunoSync.getAlunos());
    dao.close();

    Log.i("versao", preferences.getVersao());

    bus.post(new AtualizaListaAlunoEvent());
}

// ...
```

Mas na classe `AlunoPreferences` não existe esse método, então colocamos o *cursor* em cima da chamada do método e usamos o atalho "Alt + Enter" escolhemos a ***Create method 'getVersao'***. Pronto o método está criado.

Para retornar a versão persistida pelo *Shared Preferences* precisamos antes ter acesso a ao que persistiu a versão do aluno. No caso vamos ter que repetir o `SharedPreferences preferences =`

`context.getSharedPreferences(ALUNO_PREFERENCES, context.MODE_PRIVATE);` . Repetir código não é legal, podemos extrair para um método com "Ctrl + Alt + N" e na janela aberta é só clicar em "OK".

```
// ...

private SharedPreferences getSharedPreferences(){
    return context.getSharedPreferences(ALUNO_PREFERENCES, context.MODE_PRIVATE);
}

public String getVersao(){
    return null;
}
```

No método `getVersao()` vamos chamar o método `getSharedPreferences()` armazenando em uma variável chamada `preferences` , e a partir dela, chamamos o método `preferences.getString()` passando a chave `VERSAO_DO_DADO` e um segundo argumento para caso a operação não de certo, passaremos uma *string* vazia.

```
// ...

private SharedPreferences getSharedPreferences(){
    return context.getSharedPreferences(ALUNO_PREFERENCES, context.MODE_PRIVATE);
}

public String getVersao(){
    SharedPreferences preferences = getSharedPrederences();
    return preferences.getString(VERSAO_DO_DADO, "");
}
```

Agora podemos executar a nossa aplicação e verificar se tudo está funcionando. Se analisarmos o Android Monitor veremos que a aplicação sempre está buscando a versão mais recente.