02

Fazendo o envio dos dados de forma assíncrona

O envio dos dados dos contatos de nossa agenda para o servidor deverá seguir as seguintes etapas: precisamos buscar a lista no banco usando o nosso `DAO`, convertê-la em `JSON` usando o `AlunoConverter` e então enviar para o servidor por meio de uma instância de `WebClient`:

```
AlunoDAO dao = new AlunoDAO(ctx);
List<Aluno> alunos = dao.getLista();

String json = new AlunoConverter().toJSON(alunos);
String jsonDeResposta = new WebClient("http://www.meuservidor.com.br").post(json);
```

Queremos colocar no Menu de Opções o envio de dados para o servidor na opção **sincronizar**:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_novo:
            Intent intent = new Intent(ListaAlunosActivity.this, FormularioActivity.class);
            startActivity(intent);
            return false;
        case R.id.menu_enviar_alunos:
            //implementar aqui
        default:
            return super.onOptionsItemSelected(item);
        }
}
```

Basta colocarmos na linha acima a chamada da nossa classe `WebClient`, passando a lista de contatos já convertida em JSON, entretanto a execução desse código poderá ser bastante demorada, especialmente a parte na qual fazemos a requisição para o servidor. Se colocarmos esse código diretamente na `Activity`, ele será executado pela *Thread* principal do Android, a chamada ***UI Thread***. Para evitar que nossa aplicação fique "congelada" vamos executar essa tarefa em uma *Thread* separada. Teríamos algo do tipo:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_novo:
            //...
        case R.id.menu_enviar_alunos:
            new Thread(){
                public void run() {
                    AlunoDAO dao = new AlunoDAO(context);
                    List<Aluno> lista = dao.getLista();
                    dao.close();

                    String listaEmJSON = new AlunoConverter().toJSON(lista);

                    String jsonDeResposta = new WebClient("http://www.meuserver.com").post(listaEmJSON);

                    //fazer algo na View para avisar o resultado da operacao
                }
            }.start();
            return false;
        default:
            return super.onOptionsItemSelected(item);
        }
    }
}
```

Ao usar *Threads* no Android, temos que estar atentos para que todo componente de `View` seja manipulado **apenas** pela Thread principal (ou **UI Thread**), do contrário, uma *Exception* será disparada. Caso queiramos, dentro da Thread que criamos, avisar o usuário sobre o resultado da operação que executamos teríamos que fazer um código um pouco mais complicado.

Justamente para facilitar a manipulação de tarefas assíncronas, o Android possui a classe `AsyncTask`.

Vamos criar uma classe para isolar o envio de nossos contatos para um servidor remoto. Como será uma tarefa assíncrona, vamos herdar de `AsyncTask`:

```
public class EnviaContatosTask extends AsyncTask<Object, Object, String>{

}
```

No momento não se preocupe com os três tipos genéricos da classe (`<Object, Object, String>`). Por enquanto é necessário saber apenas que o terceiro tipo é referente ao retorno da operação que desejamos executar, como desejamos receber um `JSON` do servidor e ele será do tipo `String` escolhemos esse tipo.

Ao herdarmos da classe `AsyncTask` somos obrigados a sobrescrever o método `doInBackground` no qual colocaremos o conteúdo que desejamos que seja executado por uma *Thread* em paralelo:

```
public class EnviaContatosTask extends AsyncTask<Object, Object, String>{

    @Override
    protected String doInBackground(Object... params) {
        //busca da lista de alunos, conversao da lista em JSON
        //envio para o servidor
    }
}
```

Repare o tipo do parâmetro do método `doInBackground` : **Object...** trata-se de um `var-args` que permite a passagem de zero ou mais instâncias de `Object` para o método. No momento não faremos uso de recebimento de parâmetros.

Se quisermos disponibilizar nossa tarefa para que a Dalvik (ou o ART) a execute, basta invocarmos o método `execute` que herdamos da classe `AsyncTask` :

```
new EnviaContatosTask().execute();
```

Para lidarmos com o resultado da tarefa que rodou em background podemos sobrescrever o método `onPostExecute` . O tipo que o método recebe como parâmetro é o mesmo tipo de retorno do método `doInBackground` :

```
public class EnviaContatosTask extends AsyncTask<Object, Object, String> {

    @Override
    protected String doInBackground(Object... params) {
        //busca da lista de alunos, conversao da lista em JSON
        //envio para o servidor
    }

    protected void onPostExecute(String result) {
        //lidamos com o resultado do doInBackground
        //alterando o estado da tela
    }
}
```

Caso seja necessário executar algo na tela antes de começar a tarefa assíncrona podemos sobrescrever o método `onPreExecute` :

```
public class EnviaContatosTask extends AsyncTask<Object, Object, String> {

    @Override
    protected void onPreExecute() {
        //criar e alterar coisas na tela antes de executar a tarefa lenta
    }
    //...
}
```

Nossa classe `EnviaContatosTask` terá uma constante que será o endereço para o qual mandaremos o nosso JSON. No site da Caelum, temos um servidor esperando este dado, segue a constante na classe:

```
private final String endereco = "http://www.caelum.com.br/mobile" ;
```

Agora, para fazer a requisição HTTP, vamos implementar um método que usa o `HttpClient` . Vamos enviar o dado utilizando o método POST, já que uma lista tende a ser grande e pode superar o limite de caracteres do envio por GET. Após o envio, vamos abrir o que o servidor tem para nos mostrar num `Toast` .

Para que você tenha acesso à internet você deverá dar permissão para a sua aplicação:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Este trecho de código deverá estar dentro da tag `<manifest>` no seu AndroidManifest.xml.

Uma barra de progresso: ProgressDialog

Sempre que uma aplicação está esperando algum processamento que está em segundo plano, é importante avisar ao usuário. Do contrário, ele pode achar que a aplicação travou. Para não correrrmos este risco com o nosso sistema, vamos colocar uma barra de progresso. Não precisamos nos preocupar com seu xml, pois ela será inserida sobre a tela atual.

```
ProgressDialog progress = ProgressDialog.show(ListaAlunosActivity.this,
                                    "Aguarde...", "Enviando dados para a web!!!", true);
```