Conversão e envio dos dados

Nós transformamos os dados dos alunos em um json. Agora, queremos enviar os dados para um servidor.

Para o nosso curso, teremos um servidor já pronto, que está esperando este tipo de mensagem com um formato específico. Ele irá calcular a média e depois, devolver no formato json.

Com o objetivo de nos comunicarmos com o servidor, faremos uma requisição HTTP. Temoso duas formas de fazer isto: usando o método GET e o POST. Se usamos o método GET, todos os parâmetros terão que ir com a url do servidor, seguido pelos parâmetro chave e valor. Isto tem um certo limite, a partir de um certo tamanho (2KB, por exemplo) não será enviado o excedente. Para este tipo de caso, temos o método POST, no qual enviamos os dados no corpo da requisição. Neste caso, já conseguiriamos mandar toda nossa lista sem problema.

Um outro fator é que quando utilizamos o GET, ele está pedindo o dado que já está pronto no servidor. No nosso caso, estamos enviando os dados para o servidor, que irá manipula-los e depois, devolver a resposta. Nestes casos, o recomendado é usar o método POST.

Nós podemos querer fazer requisições de várias partes do nosso código, mas o ideal é criarmos uma classe para centralizarmos o acesso ao servidor remoto. Faremos isto. Criaremos a uma nova classe, que chamaremos de WebClient, que será responsável por fazer os POSTS que serão enviadas para o servidor. Por isso, iremos adicionar o método post. Iremos informar ainda que os dados enviados são em formato json.

No Android, quando quisermos fazer um conexão, usaremos a classe HttpURLConnection. Também iremos criar uma URL e adicionaremos o endereço do nosso servidor. Temos um servidor que já está esperando nossas requisições: https://www.caelum.com.br/mobile. Depois, usando o comando ALT + ENTER, selecionaremos o "Surround with try/catch". Colocaremos o HttpURLConnection dentro do "try/catch".

public class WebClient {  
 public void post(String json) {  
 try {  
 URL url = new URL("https://www.caelum.com.br/mobile");  
 HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
  
 } catch (MalformedURLException e) {  
 e.prinStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 }

Para construirmos um objeto connection, vamos pedir para a url abrir conexão com a url que especificamos mais acima. Então, vamos evocar o método openConnection no objeto url. Observe que adicionamos mais uma cláusula catch dentro do "try/catch". Com a últimas alterações, a openConnection irá nos devolver uma url. Nós fizemos um cast para HttpURLConnectione temos a conexão pronta.

Nós queremos fazer um método POST com esta conexão, para isto usaremos o método setDoOutPut. Se estivéssemos trabalhando com um SET, isto não seria necessário, porque não precisaríamos de saída. Mas no nosso caso, passaremos o parâmetro true.

Também iremos adicionar PrintStream, passando como parâmetro getOutputStream que tiramos da conexão.

public class WebClient {  
 public void post(String json) {  
 try {  
 URL url = new URL("https://www.caelum.com.br/mobile");  
 HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
  
 connection.setDoOutPut(true);  
  
 new PrintStream(connection.getOutputStream());   
 } catch (MalformedURLException e) {  
 e.prinStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }

Vamos atribuí-la a uma variável local, que iremos chamar de output. Depois, faremos adicionaremos output.println() e vamos escrever no Streamde saída, o json que representa os alunos.

Em seguida, iremos enviar para o servidor e receber a resposta. Para isto, a primeira coisa que teremos que fazer é adicionar connection.connect e assim, conectarmos com o servidor. Agora, precisamos da resposta do mesmo. O próprio objeto connection tem InputStream, com o qual podemos ler a resposta. Vamos criar um Scanner, que irá facilitar a leitura, e passaremos como parâmetro o Stream de entrada.

public class WebClient {  
 public void post(String json) {  
 try {  
 URL url = new URL("https://www.caelum.com.br/mobile");  
 HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
  
 connection.setDoOutPut(true);  
  
 PrintStream output = new PrintStream(connection.getOutputStream());  
 output.println(json);  
  
  
 Scanner scanner = new Scanner(connection.getOutputStream());  
 } catch (MalformedURLException e) {  
 e.prinStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}  
  
}

Para lermos a resposta, vamos criar String resposta e chamaremos o método next. Em seguida, iremos mudar o valor de retorno para o usuário: onde está public void alteraremos para public String. A String que iremos devolver será a resposta que lemos do servidor. Iremos adicionar também um return resposta.

public class WebClient {  
 public String post(String json) {  
 try {  
 URL url = new URL("https://www.caelum.com.br/mobile");  
 HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
  
  
 connection.setDoOutPut(true);  
  
 PrintStream output = new PrintStream(connection.getOutputStream());  
 output.println(json);  
  
 Scanner scanner = new Scanner(connection.getOutputStream());  
 String resposta = scanner.next();  
 return resposta;  
 } catch (MalformedURLException e) {  
 e.prinStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}  
}

Até aqui, pensamos na possibilidade de que tudo saiu como esperado. Mas se acontecer alguma exception, ele irá cair em um dos catchs e irá imprimir a StackTrace. Mas precisamos avisar quem estiver usando a classe WebDClient de que ocorreu um problema. Para isto, iremos adicionar return null.

\...  
  
 Scanner scanner = new Scanner(connection.getOutputStream()); String  
 resposta = scanner.next(); return resposta;  
} catch (MalformedURLException e) {  
 e.prinStackTrace();  
} catch (IOException e) {  
 e.printStackTrace();  
}  
return null;  
\...

Agora, precisamos cuidar de um detalhe. O nosso servidor estará esperando os dados no formato json e nós esperamos que a resposta venha da mesma forma. Seria interessante especificarmos isto na requisição. Precisaremos adicionar estas informações no cabelho da requisição. Usaremos os método setRequestProperty e depois, as informações extras: Content-type e application/json. Faremos uma linha parecida, mas referente à resposta.

public class WebClient {  
 public String post(String json) {  
 try {  
 URL url = new URL("https://www.caelum.com.br/mobile");  
 HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
 connection.setRequestProperty("Content-type", "application/json");  
 connection.setRequestProperty("Accept", "application/json")  
\...

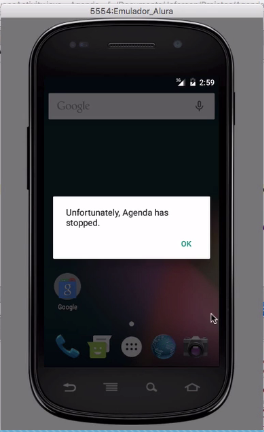
A classe WebClient está pronta, só falta utilizá-la. Voltaremos para o ListaAlunosActivity, em que convertíamos os alunos para json.

@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
 switch (item.getItemId()) {  
 case R.id.menu\_enviar\_notas:  
 AlunoDAO dao = new AlunoDAO(this);  
 List<Aluno> alunos = dao.buscaAlunos();  
 dao.close();  
  
 String json = conversor.converteParaJSON(alunos);  
  
 Toast.makeText(this, "Enviando notas...", Toast.LENGHT\_LONG).show();  
 break;  
 }  
}

Antes do Toast, vamos criar a classe WebClient, em seguida, adicionaremos o método client.post. Como parâmetro, usaremos o json. Quando chamamos o post, ele devolver a tarefa que o servidor nos enviou, iremos criar a variável String reposta que pega a resposta. E iremos adicioná-la também no método Toast.

@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
 switch (item.getItemId()) {  
 case R.id.menu\_enviar\_notas:  
 AlunoDAO dao = new AlunoDAO(this);  
 List<Aluno> alunos = dao.buscaAlunos();  
 dao.close();  
  
 String json = conversor.converteParaJSON(alunos);  
  
 WebClient client = new WebClient();  
 String resposta = client.post(json);  
 Toast.makeText(this, resposta, Toast.LENGHT\_LONG).show();  
 break;  
 }  
}

Vamos ver o que acontecerá se fizermos um novo teste no emulador e clicarmos no ícone do menu.



Recebemos a mensagem de que "a Agenda parou". Sabemos que quando isto acontece é porque ocorreu uma exception. Se pesquisarmos, descobriremos que ocorreu uma NetworkOnMainThreadException. Ele está tentando fazer uma operação de rede na thread principal... Veremos o que isto significa e iremos resolver o problema mais adiante. Mas por enquanto, não poderemos fazer o envio para o servidor. Falta resolvermos apenas um detalhe.