4. A konténerek

A konténer arra a célra szolgál, hogy más konténereket vagy komponenseket tegyünk bele, ezzel megszabva az elhelyezkedés pontos és kevésbé pontos szabályait. Szinte kivétel nélkül fa struktúrába szervezett konténerekbl állítjuk össze a felhasználói interfészt, s az elz fejezetben részletezett komponensek lesznek a fa levelei. A mobil eszköz kis képernyje és az ablakok kötött alakja okán nem kell attól tartanunk, hogy túl sok konténer használatát kellene megtanulnunk, alig pár konténer típussal fogunk találkozni, ám ezekkel minden szükséges mveletet meg tudunk tenni.

4.1. Konténerek

4.1.1. LinearLayout

Az eddigi példákban szinte kivétel nélkül a *LinearLayout* szerepelt, mivel ez olyan egyszer, mint a faék: a hozzáadott komponenseket egymás mellé vagy egymás alá pakolja. Ha túl sok komponenst adunk hozzá, akkor egyszeren nem ábrázolja a felesleget, azok kicsúsznak a kijelzrl. Mindegyik *LinearLayout* példánynak megvan a maga *gravitációja*, alapesetben balra és felfelé "esnek" a komponenseink, az els komponensünk a bal fels sarokba fog kerülni.

Nézzünk egy egyszer példát, amelyben egy felhasználónevet és jelszót bekér képernytervet írunk le XML alapokon. Ehhez a *res* mappában a *layout* alatti *main.xml* állomány tartalmát meg kell változtatnunk:

main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:background="#000044"
    android:orientation="vertical">
  <TextView android:id="@+main/usernameLabel"
        android:text="@string/usernameLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#ffffff" />
  <EditText android:id="@+main/usernameField"
        android:text="@string/usernameField"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
  <TextView android:id="@+main/passwordLabel"
        android:text="@string/passwordLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#ffffff" />
  <EditText android:id="@+main/passwordField"
        android:text="@string/passwordField"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:password="true" />
  <Button android:id="@+main/signInButton"
        android:text="@string/signInLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Láthatjuk, hogy a *LinearLayout* fogja közre a többi komponenst, amelyeket egymás alá szeretnénk helyezni. Nézzük meg közelebbrl a *LinearLayout*attribútumait:

- **android:layout_height**, a konténer magassága. Értéke lehet *fill_parent*, amikor kitölti a szül konténer magasságát; *wrap_content*, amikor a benne lév egyéb komponensek magasságát veszi fel; lehet pixelben megadott érték (például: *200px*), ekkor pont ezt a méretet veszi fel.
- **android:layout_width**, a konténer szélessége. Azonosan mködik, mint a magasság.
- **android:orientation**, a konténer feltöltési iránya. Lehet *horizontal*, amikor egymás mellé pakolja a komponenseket, illetve lehet *vertical*, amikor egymás alá. A vízszintes feltöltés az alapértelmezett.

A komponensek esetén is hasonlóképp értelmezhet a *layout_height* és a *layout_width*. Vegyük észre az *android:text* attribútumokat, ahol hivatkozunk egyéb erforrásokra is, ezeket fel kell vennünk a *strings.xml* állományban:

strings.xml

```
<resources>
    <string name="app_name">HelloJavaForum</string>
    <string name="usernameLabel">Username:</string>
    <string name="usernameField">Username</string>
    <string name="passwordLabel">Password:</string>
    <string name="passwordField">Password</string>
    <string name="signInLabel">Sign in!</string>
</resources>
```

A használathoz egyszeren be kell állítanunk a létrehozott erforrást, mint képernykép:

Java

```
super.onCreate(savedInstanceState);

try
{
  setContentView(R.layout.main);
} catch (Exception except)
{
  TextView textView = new TextView(this);
  final Writer result = new StringWriter();
  final PrintWriter printWriter = new PrintWriter(result);
  except.printStackTrace(printWriter);
  textView.setText(result.toString());
  setContentView(textView);
}
```

A kijelzn pedig megjelenik a várt elrendezés:

4.1.2. TableLayout és TableRow

Az esetek nagy részében táblázatos formában szeretnénk komponenseket kitenni a képernyre. A *TableLayout* esetén táblázat celláiba kerülnek komponensek és a *TableRow* határoz meg egy-egy sort a táblázatban:

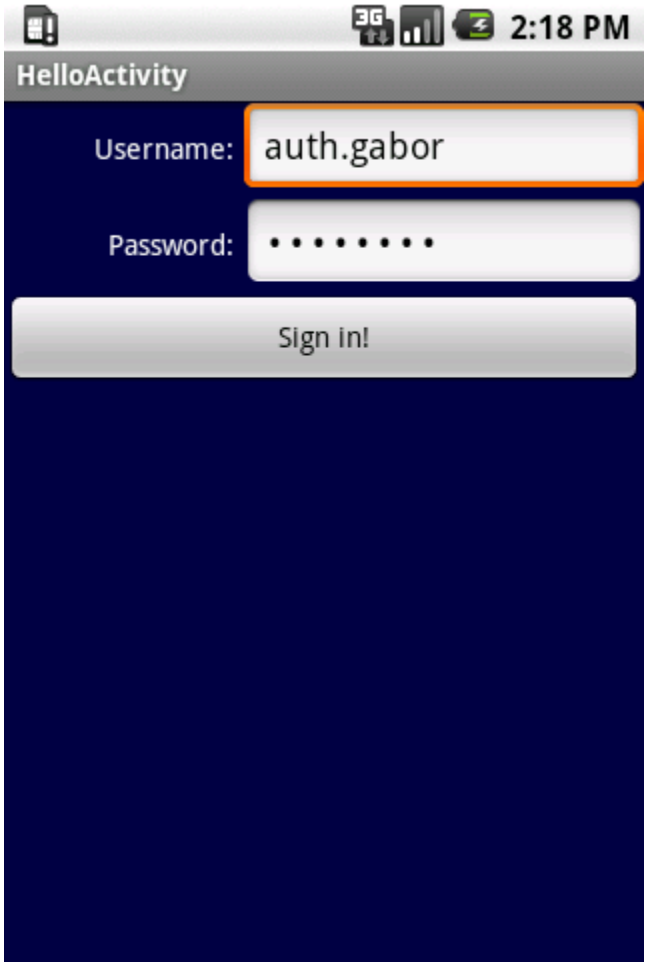
main.xml

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:background="#000044">
  <TableRow>
    <TextView android:id="@+main/usernameLabel"
              android:text="@string/usernameLabel"
              android:textColor="#ffffff"
              android:gravity="right"
              android:paddingRight="5px"
              android:layout_weight="1"
              android:layout_width="fill_parent"/>
    <EditText android:id="@+main/usernameField"
              android:text="@string/usernameField"
              android:layout_weight="2"
              android:layout_width="fill_parent" />
  </TableRow>
  <TableRow>
    <TextView android:id="@+main/passwordLabel"
              android:text="@string/passwordLabel"
              android:textColor="#ffffff"
              android:gravity="right"
              android:paddingRight="5px"
              android:layout_weight="1"
              android:layout_width="fill_parent"/>
    <EditText android:id="@+main/passwordField"
              android:text="@string/passwordField"
              android:password="true"
              android:layout_weight="2"
              android:layout_width="fill_parent"/>
  </TableRow>
  <TableRow>
    <Button android:id="@+main/signInButton"
            android:text="@string/signInLabel"
            android:layout_span="2"
            android:layout_weight="2"/>
  </TableRow>
</TableLayout>
```

Az XML tartalmaz néhány újabb attribútumot, amely ismeretlen lehet:

- **android:gravity**, az igazodás iránya a rendelkezésre álló cellában. Értéke lehet *left*, *center_horizontal* illetve *right*, amelyek balra, középre vagy jobbra igazítják a komponenst.
- **android:padding**, a komponens körül hagyott üres hely. Az Android a komponenseket szorosan egymás mellé igazítja, amely néha rondán néz ki, ilyen esetben meg kell adnunk *padding* értéket. Ha nem szeretnénk minden oldalra azonos méret helyet, akkor használhatjuk a *paddingLeft*, a *paddingTop*, a *paddingRight*, illetve a *paddingBottom* attribútumokat.
- **android:layout_weight**, a komponens *súlya*. Ha egymás mellé teszünk több komponenst azzal az utasítással, hogy töltsék ki a szabad helyet, akkor az Android egyenl arányban osztja ki rendelkezésre álló területet. Ezt tudjuk befolyásolni súlyozással: amelyik komponensnek nagyobb súlyt adunk, az fog több helyet elfoglalni. A fenti példában a beviteli mezk nagyobb súlyt kapnak a címkékkel szemben.
- **android:layout_span**, az elfoglalt cellák száma. Ha egy sorba kevesebb komponens kerül, mint amennyi cella adódik a táblázatban, akkor ezzel az attribútummal jelezhetjük, hogy a komponens több cellába is terjeszkedhet.

A *TableLayout* táblázata annyi sort tartalmaz, amennyi *TableRow* meg van adva az XML-ben, illetve annyi oszlopa van, amennyi adódik a komponensek számából. Nézzük az eredményt:



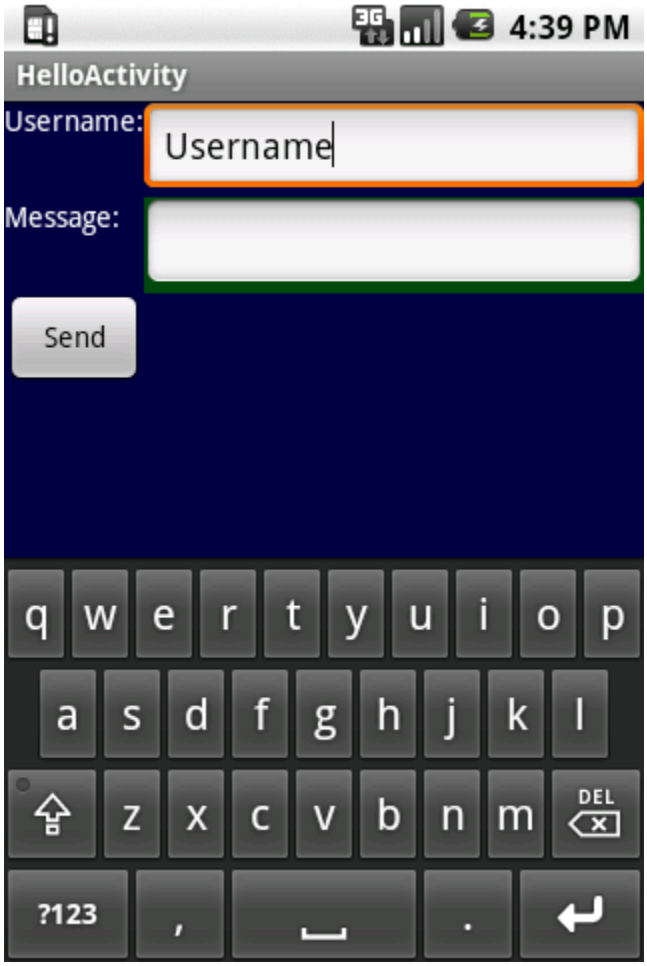
4.1.3. ScrollView

Ha olyan programot fejlesztünk, amelynek a komponensei nem férnek el a kijelzn, akkor egyszeren bele kell tennünk egy *ScrollView* konténerbe, és a platform gondoskodik a görgetésrl:

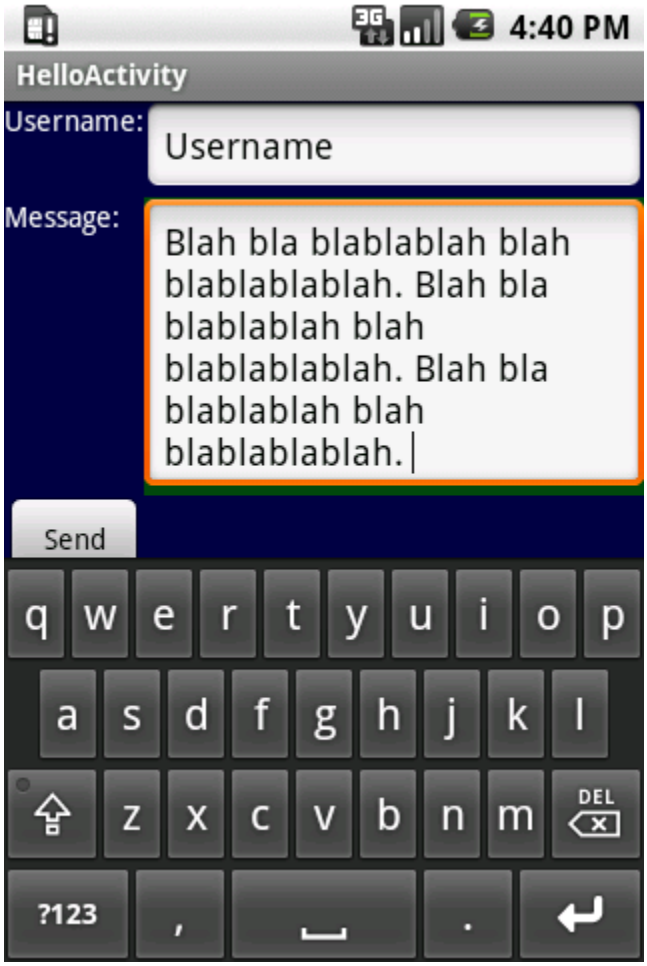
main.xml

```
 <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+main/messageLayout"
    android:background="#000044"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent">
  <TableRow>
    <TextView android:id="@+main/usernameLabel"
              android:text="@string/usernameLabel"
              android:textColor="#ffffff"
              android:layout_height="fill_parent"
              android:layout_width="fill_parent"/>
    <EditText android:id="@+main/usernameField"
              android:text="@string/usernameField"
              android:layout_height="fill_parent"
              android:layout_width="fill_parent"
              android:layout_weight="1"/>
  </TableRow>
  <TableRow>
    <TextView android:id="@+main/messageLabel"
              android:text="@string/messageLabel"
              android:textColor="#ffffff"
              android:layout_height="fill_parent"
              android:layout_width="fill_parent"/>
    <ScrollView android:id="@+main/messageScrollView"
                android:background="#004400"
                android:layout_height="fill_parent"
                android:layout_width="fill_parent"
                android:layout_weight="1">
      <EditText android:id="@+main/messageField"
                android:layout_height="fill_parent"
                android:padding="10px"
                android:layout_width="fill_parent"
                android:layout_weight="1"/>
    </ScrollView>
  </TableRow>
  <TableRow>
    <Button android:id="@+main/sendButton"
            android:text="@string/sendLabel" />
  </TableRow>
</TableLayout>
```

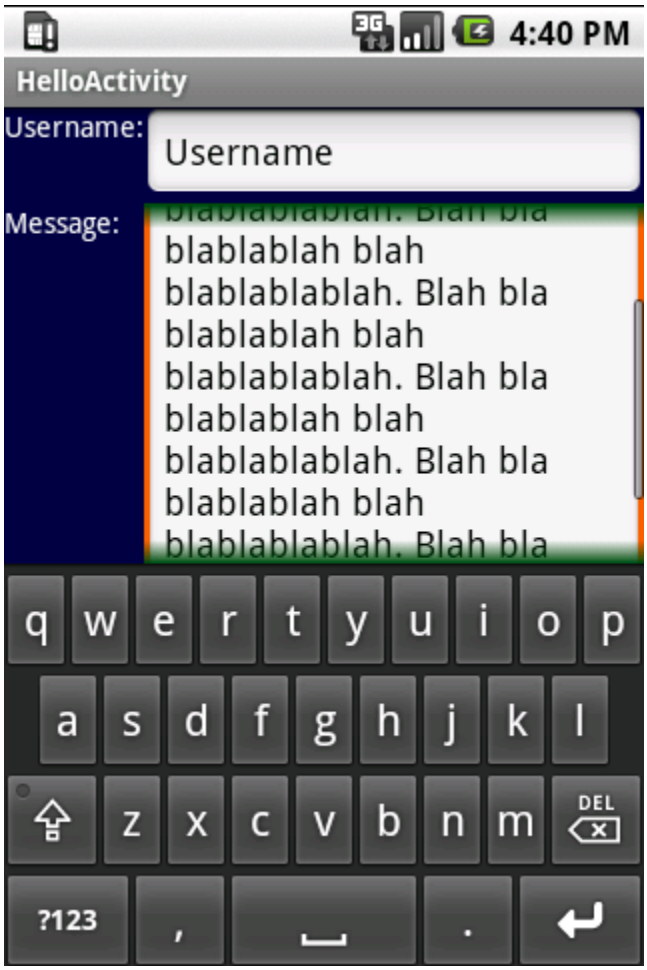
Figyeljük meg, hogy sehol nem adtunk meg konkrét méretet, ellenben a komponensek súlyát meghatároztuk: erre érdemes ügyelni, mivel az Android platform változatos képernyméreteken is képes futni, így nem tudhatjuk elre, mekkora helyet kell lefoglalnunk az alkalmazásunk számára. Ha van némi idnk, akkor célszer több elterjedt méretben megtekinteni a programunk ablakait. Ha elindítjuk a fenti *layout*programját, akkor az alábbi képernyt kell látnunk:



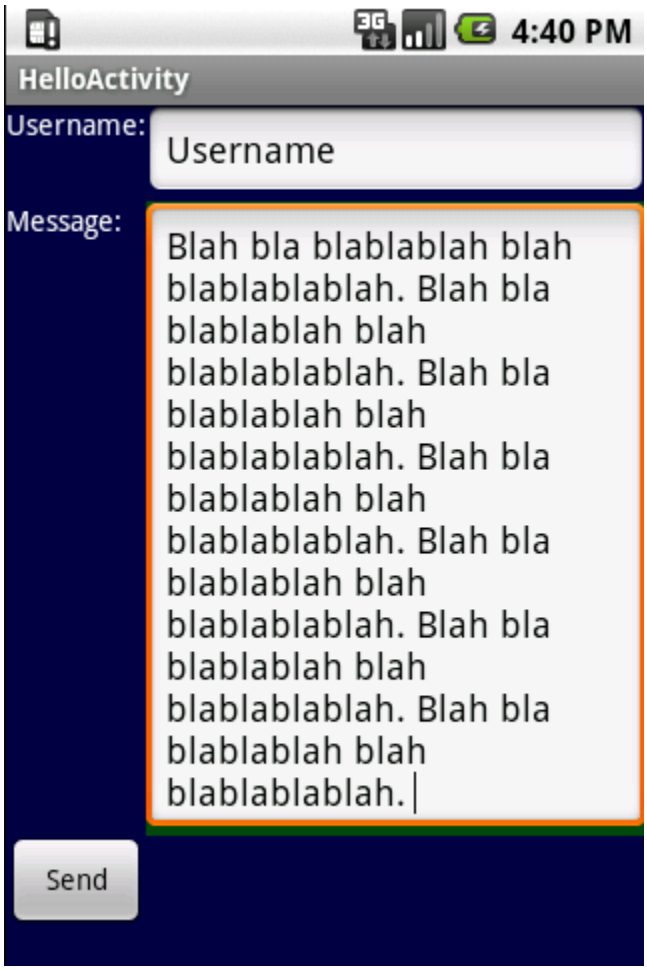
Pont úgy néz ki, mint ahogy szokott, egyedül az üzenet beviteli mezje lehet fura, mivel zöld a háttere, ugyanis a *ScrollView*hátterét zöldre színeztük. Kezdjünk bele írni:



A szövegbeviteli mez elkezd hízni és kezdi felzabálni a rendelkezésre álló helyet, a *Send* gomb lassan eltnik a virtuális billentyzet mögött, ám görget sávnak még nyoma nincs. Ez a viselkedés a súlyozásból ered, mind az *EditText*, mind a *ScrollView*súlyozott komponens, tehát a többi rovására fog növekedni, ám a képerny határait elérve megjelenik a görget sáv:



Ahogy látszik, a beviteli mez egészen a virtuális billentyzetig növekszik, majd görgethetvé válik a tartalma. Ha eltntetjük a virtuális billentyzetet, akkor a beviteli mez fel tudja venni a keletkez helyet és el is tnik a görget sáv:



4.1.4. TabHost és TabWidget

Ha egy képernyn nem fér el minden beviteli mez, akkor érdemes a füles konténerhez nyúlni, amely lehetvé teszi, hogy a felhasználó egy Activity-n belül több képernynyi bevitelt végezhessen el. Kissé nehézkessé sikeredett ez a konténer, de idvel talán egyszerbb lesz a használata. Nézzünk példát egy olyan képernytervre, ahol meg kell adni nevet jelszót és opcionálisan a proxy szervert:

main.xml

```
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
         android:id="@+main/tabhost"
         android:layout_height="fill_parent"
         android:layout_width="fill_parent">
  <TabWidget android:id="@android:id/tabs"
             android:layout_height="wrap_content"
             android:layout_width="fill_parent"/>
  <FrameLayout android:id="@android:id/tabcontent"
               android:paddingTop="60px"
               android:layout_height="fill_parent"
               android:layout_width="fill_parent">
    <TableLayout android:id="@+main/loginTable"
                 android:layout_height="fill_parent"
                 android:layout_width="fill_parent"
                 android:background="#000044">
      <TableRow>
        <TextView android:id="@+main/usernameLabel"
                  android:text="@string/usernameLabel"
                  android:textColor="#ffffff"
                  android:gravity="right"
                  android:paddingRight="5px"
                  android:layout_weight="1"
                  android:layout_width="fill_parent"/>
```

```
        <EditText android:id="@+main/usernameField"
                  android:text="@string/usernameField"
                  android:layout_weight="2"
                  android:layout_width="fill_parent" />
      </TableRow>
      <TableRow>
        <TextView android:id="@+main/passwordLabel"
                  android:text="@string/passwordLabel"
                  android:textColor="#ffffff"
                  android:gravity="right"
                  android:paddingRight="5px"
                  android:layout_weight="1"
                  android:layout_width="fill_parent"/>
        <EditText android:id="@+main/passwordField"
                  android:text="@string/passwordField"
                  android:password="true"
                  android:layout_weight="2"
                  android:layout_width="fill_parent"/>
      </TableRow>
      <TableRow>
        <Button android:id="@+main/signInButton"
                android:text="@string/signInLabel"
                android:layout_span="2"
                android:layout_weight="2"/>
      </TableRow>
    </TableLayout>
    <TableLayout android:id="@+main/proxyTable"
                 android:layout_height="fill_parent"
                 android:layout_width="fill_parent"
                 android:background="#000044">
      <TableRow>
        <TextView android:id="@+main/hostLabel"
                  android:text="@string/hostLabel"
                  android:textColor="#ffffff"
                  android:gravity="right"
                  android:paddingRight="5px"
                  android:layout_weight="1"
                  android:layout_width="fill_parent"/>
        <EditText android:id="@+main/hostField"
                  android:text="@string/hostField"
                  android:layout_weight="2"
                  android:layout_width="fill_parent" />
      </TableRow>
      <TableRow>
        <TextView android:id="@+main/portLabel"
                  android:text="@string/portLabel"
                  android:textColor="#ffffff"
                  android:gravity="right"
                  android:paddingRight="5px"
                  android:layout_weight="1"
                  android:layout_width="fill_parent"/>
        <EditText android:id="@+main/portField"
                  android:text="@string/portField"
                  android:password="true"
                  android:layout_weight="2"
                  android:layout_width="fill_parent"/>
      </TableRow>
      <TableRow>
        <CheckBox android:id="@+main/useIt"
                  android:text="@string/useItLabel"
                  android:layout_span="2"
                  android:layout_weight="2"/>
      </TableRow>
    </TableLayout>
  </FrameLayout>
</TabHost>
```

A *TabHost* konténer fogja közre azt, amit szeretnénk a fülekben használni, benne egy *TabWidget* adja a fülek rajzolatát, s ez után egy *FrameLayout*blokkb a kell tennünk azokat a komponenseket, amelyeket a fülekre szánunk. Három dologra kell figyelnünk:

- A *TabWidget* a *TabHost* els gyereke legyen, s az azonosítója **@android:id/tabs** kell legyen.
- A *TabWidget* után következ bejegyzés egy *FrameLayout* kell legyen **@android:id/tabcontent** azonosítóval.
- A *FrameLayout* tartalmazzon egy *paddingTop* attribútumot, amelyben a fülek magasságát adjuk meg.

Az XML után még pár sort kell írnunk a programba is:

Java

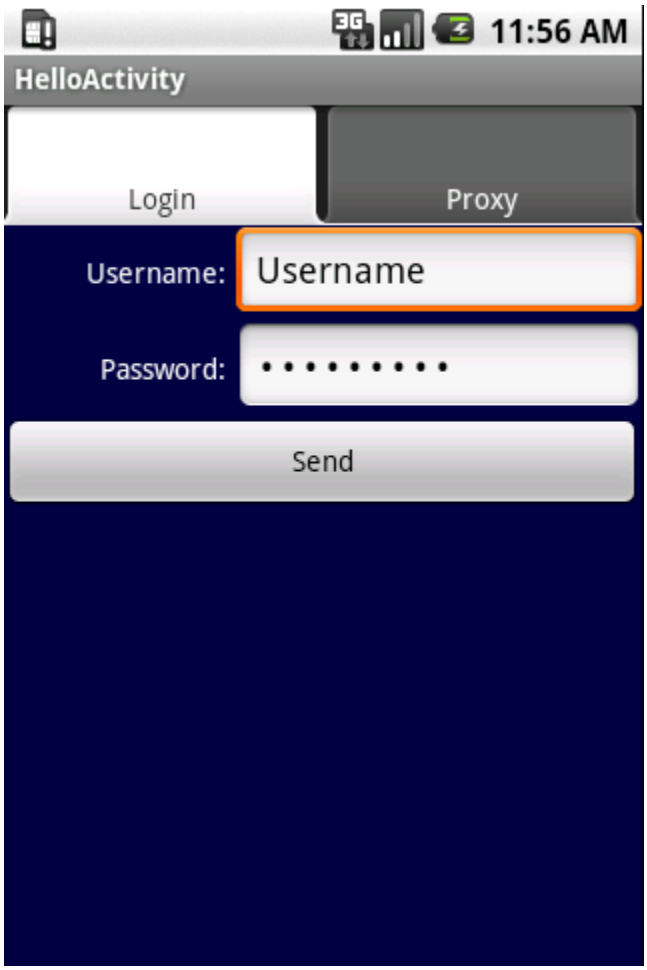
```
super.onCreate(savedInstanceState);

try
{
  setContentView(R.layout.main);

  final TabHost tabs = (TabHost) findViewById(R.main.tabhost);
  tabs.setup();

  TabHost.TabSpec loginSpec = tabs.newTabSpec("loginTable");
  loginSpec.setContent(R.main.loginTable);
  loginSpec.setIndicator("Login");
  tabs.addTab(loginSpec);

  TabHost.TabSpec proxySpec = tabs.newTabSpec("proxyTable");
  proxySpec.setContent(R.main.proxyTable);
  proxySpec.setIndicator("Proxy");
  tabs.addTab(proxySpec);
  tabs.setCurrentTab(0);
} catch (Exception except)
{
  TextView textView = new TextView(this);
  final Writer result = new StringWriter();
  final PrintWriter printWriter = new PrintWriter(result);
  except.printStackTrace(printWriter);
  textView.setText(result.toString());
  setContentView(textView);
}
```

Mindezek után a szövegek közé fel kell vennünk az új hivatkozásokat:

strings.xml

```
 <resources>
    <string name="app_name">HelloJavaForum</string>
    <string name="usernameLabel">Username:</string>
    <string name="usernameField">Username</string>
    <string name="passwordLabel">Password:</string>
    <string name="passwordField">Type here</string>
    <string name="signInLabel">Send</string>
    <string name="hostLabel">Host:</string>
    <string name="hostField"></string>
    <string name="portLabel">Port:</string>
    <string name="portField"></string>
    <string name="useItLabel">Use proxy</string>
</resources>
```

S már kész is vagyunk, lássuk ezt mködés közben:



4.1.5. Egyéb konténerek

Létezik még néhány konténer, amelyek közül tudunk válogatni:

- **FrameLayout**, amely olyan, mint egy kártyapakli, az újonnan hozzáadott elem eltakarja a régebbieket (lásd elz fejezet).
- **RelativeLayout**, amely esetén egy már meglév komponenshez tudjuk hozzáragasztani az újabbat.
- **AbsoluteLayout**, amelynél pontosan meg kell mondanunk, hogy a komponensek hova kerüljenek és mekkora helyet foglaljanak.