Andriod alapú szoftverfejlesztés

2. Labor

Felhasználói felület tervezés és készítés Android platformon

Tartalom

1 Felkészülés a mérésre

A mérés célja a felhasználói felület tervezésének és készítésének bemutatása Android platformon. A mérés elején a laborvezető röviden bemutatja a *ListActivity* használatát, ezt követően pedig egy Todo lista alkalmazást készítünk.

A mérés az alábbi témákat érinti:
- *ListActivity* és *ListAdapter* használata
- *RelativeLayout* és *TableLayout*
- Alap vezérlők: *TextView*, *Button*, *EditText*, *Spinner*
- Dátumválasztó dialógus

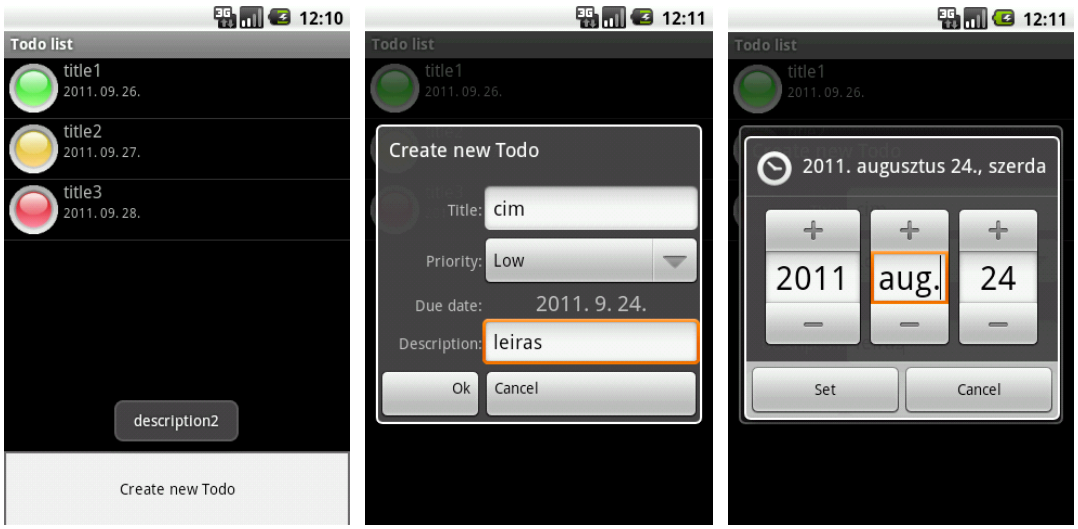
2 ListActivity használata

A `ListActivity` osztály tipikusan egy teljes képernyős lista megjelenítésekor használatos. Az Activity-ktől megszokott képességeken kívül a `ListActivity` alapértelmezetten tartalmaz egy `ListView`-t, melyet a `getListView()` függvénnyel kérhetünk el és egy `ListAdapter`-t, mely a lista elemeinek tárolásáért és megjelenítésért felelős. A `ListAdapter` tipikusan egy `BaseAdapter`-ből leszármazó osztály, mely tartalmazza az elemeket (objektumok listája), felelős új elem hozzáadásáért, eléréséért és törléséért, valamint a `getView(…)` függvény felüldefiniálásával megadhatjuk, hogy a lista egy sorában egy elem hogyan renderlődjön ki. A `getView(…)` függvény paraméterül kap egy kirenderelendő objektumot (pl. egy Todo objektumot). A `getView(…)` függvényben tehát tipikusan kiválasztunk egy XML-ben leírt elrendezést a sorra és beállítjuk az abban lévő felületi elemek tartalmát, képeket. stb. Ezáltal tehát megadhatjuk, hogy a lista elemei hogyan nézzenek ki és nem csak standard egysoros listákat hozhatunk létre, hanem gyakorlatilag korlátlan szabadsággal rendelkezünk.

3 Laborfeladatok

A labor során egy Todo lista alkalmazást hozunk létre. Az alkalmazás indításakor egy `ListActivity` jelenik meg, mely tartalmazza a *Todo* elemeket. Egy *Todo*-ról az alábbiakat tároljuk:
- cím (title)
- prioritás (LOW, MEDIUM, HIGH)
- esedékesség dátuma (duedate)
- leírás (description

Egy listaelem kinézete a következő: bal oldalt a prioritásnak megfelelő ikon látszik, jobb oldalt pedig felül a *Todo* címe, alatta pedig kisebb betűvel az esedékesség dátuma.
Egy listaelemre röviden kattintva (`getListView().setOnItemClickListener(…)`) a *Todo* elem leírása jelenik meg egy `Toast` ablakban.
Egy listaelemre hosszan kattintva (`registerForContextMenu(getListView());`) egy helyi menü jelenik meg *Delete* és *Back* gombokkal. A *Delete* gombot választva törlődik a kiválasztott listaelem.
Amennyiben az idő engedi, folytassuk a munkát a következő *Bonus* feladattal: A `ListActivity` egy saját menüvel rendelkezik, melyben egy „Create new Todo” menüpont található, melyet kiválasztva dialógus formában egy új `Activity` jelenik meg, ahol egy `TableLayout` elrendezésen a létrehozandó új *Todo* adatait adhatjuk meg. Az új *Todo* esedékességi dátumát egy dátumválasztó dialógussal (`DatePickerDialog`) oldjuk meg.
A labor során többször lesz lehetőség a forráskódot a dokumentumból másolni, azonban PDF nem ideális erre, ezért a szükséges kódrészek a labor anyagban található patch.txt-ből elérhetők.



Todo alkalmazás

3.1 ListActivity létrehozása

Első lépésként hozzuk létre a listát és töltsük fel három minta elemmel. A kezdő `Activity` egy `ListActivity` lesz, melyet a 3.1.4-ben definiálunk.

3.1.1 Szöveg erőforrás létrehozása (values/strings.xml)

Helyezzük el a values/strings.xml-be az alábbi tartalmat, a labor során szükség lesz erre az erőforrásra:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Todo list</string>

    <string name="itemCreateTodo">Create new Todo</string>

    <string name="lblTodoTitle">Title:</string>
    <string name="lblTodoPriority">Priority:</string>
    <string name="lblTodoDueDate">Due date:</string>
    <string name="lblTodoDescription">Description:</string>

    <string name="btnOk">Ok</string>
    <string name="btnCancel">Cancel</string>
</resources>
```

3.1.2 Todo osztály

Hozzuk létre a `Todo` osztályt, ezt példányosítva fogjuk „bean” modellhez hasonlóan tárolni a Todo elemeket. Figyeljük meg a prioritásokat leíró Priority enumerációt.

```
public class Todo {

     public enum Priority { LOW, MEDIUM, HIGH }

     private String title;
     private Priority priority;
     private String dueDate;
     private String description;

     public   Todo(String   aTitle,   Priority   aPriority,
       String aDueDate, String aDescription)
     {
          title = aTitle;
          priority = aPriority;
          dueDate = aDueDate;
          description = aDescription;
     }
```

```
    public String getTitle() {
        return title;
    }

    public Priority getPriority() {
        return priority;
    }

    public String getDueDate() {
        return dueDate;
    }

    public String getDescription() {
        return description;
    }
}
```

3.1.3 TodoAdapter osztály létrehozása

Hozzunk létre egy `TodoAdapter` osztályt, amely a `BaseAdapter`-ből öröklődik. Az osztály feladata a létrehozott *Todo* elemek tárolása, kezelése és a listában való megjelenítési módjuk megadása a `getView(…)` függvényben.

```
public class TodoAdapter extends BaseAdapter {

    private final List<Todo> todos;

    public TodoAdapter(final Context context, final ArrayList<Todo>
aTodos) {
      todos = aTodos;
    }

    public void addItem(Todo aTodo)
    {
      todos.add(aTodo);
    }

    public int getCount() {
        return todos.size();
    }

    public Object getItem(int position) {
        return todos.get(position);
    }

    public long getItemId(int position) {
        return position;
    }
```

```
    // Sor megjelenenítésének beállítása
    public View getView(int position, View convertView, ViewGroup
parent) {

        final Todo todo = todos.get(position);

        LayoutInflater inflater = (LayoutInflater)
parent.getContext().getSystemService(
            Context.LAYOUT_INFLATER_SERVICE);
        View itemView = inflater.inflate(R.layout.todorow, null);

        ImageView imageViewIcon = (ImageView)
itemView.findViewById(R.id.imageViewPriority);
        switch (todo.getPriority()) {
                  case LOW:
                        imageViewIcon.setImageResource(R.drawable.low);
                        break;
                  case MEDIUM:

      imageViewIcon.setImageResource(R.drawable.medium);
                        break;
                  case HIGH:

      imageViewIcon.setImageResource(R.drawable.high);
                        break;
                  default:

      imageViewIcon.setImageResource(R.drawable.high);
                        break;
            }

        TextView textViewTitle = (TextView)
itemView.findViewById(R.id.textViewTitle);
        textViewTitle.setText(todo.getTitle());

        TextView textViewDueDate = (TextView)
itemView.findViewById(R.id.textViewDueDate);
        textViewDueDate.setText(todo.getDueDate());

        return itemView;
    }

    /**
     * Egye elem törlése
     */
    public void deleteRow(Todo aTodo) {
        if(todos.contains(aTodo)) {
            todos.remove(aTodo);
        }
    }
}
```

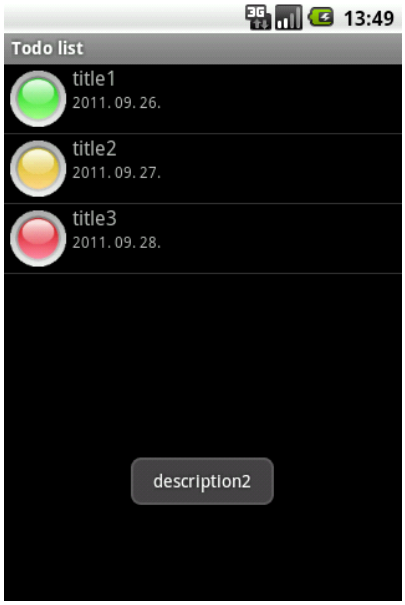
A `getView(…)` függvényben figyeljük meg, hogy állítjuk be az elrendezést `LayoutInflater` használatával. Ebben a függvényben hivatkozunk a `R.layout.todorow` erőforrásra, melynek tartalma (layout/todorow.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="horizontal"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content">

    <ImageView
     android:id="@+id/imageViewPriority"
     android:layout_height="wrap_content"
     android:layout_width="wrap_content"
     android:src="@drawable/high"
     android:padding="5dp"/>

    <RelativeLayout
     android:layout_height="wrap_content"
     android:layout_width="fill_parent">
         <TextView
          android:id="@+id/textViewTitle"
          android:textSize="16dp"
          android:text="Title"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_alignParentTop="true"/>

         <TextView
          android:id="@+id/textViewDueDate"
          android:textSize="12dp"
          android:text="DueDate"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_below="@id/textViewTitle"
          android:layout_alignParentBottom="true"
          android:gravity="bottom"/>
     </RelativeLayout>

</LinearLayout>
```

Továbbá szintén a `getView(…)` függvényben hivatkozunk három képre (`R.drawable.low/medium/high`), melyek elérhetők a todopriorities.zip állományból.

3.1.4 ListActivity definiálása és Description megjelenítése Toast-ban

A `ListActivity onCreate(…)` függvényében hozzuk létre az *Adapter*-t, adjunk hozzá néhány példa elemet, majd pedig állítsuk be a `ListActivity`-nek *Adapter*-ként. Ezután a `ListView`-nek állítsunk be egy `OnItemClickListener`-t, melyben megvalósítjuk, hogy a *Todo description* mezője jelenjen meg egy `Toast`-ban.



Todo lista

```
public class ActivityMain extends ListActivity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Adapter létrehozása és feltöltése néhány elemmel
        ArrayList<Todo> todos = new ArrayList<Todo>();
        todos.add(new Todo("title1", Priority.LOW, "2011.
09. 26.", "description1"));
        todos.add(new Todo("title2", Priority.MEDIUM,
"2011. 09. 27.", "description2"));
        todos.add(new Todo("title3", Priority.HIGH, "2011.
09. 28.", "description2"));
        TodoAdapter todoAdapter = new TodoAdapter(
                getApplicationContext(), todos);
        setListAdapter(todoAdapter);

        // Elem kattintás eseményre feliratkozás -
description megjelenítése
```

```
        getListView().setOnItemClickListener(new
OnItemClickListener() {
          public void onItemClick(AdapterView parent, View
v, int position, long id) {
               Todo selectedTodo =
(Todo)getListAdapter().getItem(position);
               Toast.makeText(ActivityMain.this,
selectedTodo.getDescription(), Toast.LENGTH_LONG).show();
          }
          });
    }
}
```

3.2 Listaelem törlése

A feladat, hogy egy listaelemre hosszan kattintva (ListActivity-ben: `registerForContextMenu(getListView());`) jelenjen meg egy helyi menü *Delete* és *Back* menüpontokkal és a *Delete*-t választva törlődjön a kiválasztott elem.
Adjuk az alábbi sort a ListActivity onCreate(…) függvényének végéhez:

```
registerForContextMenu(getListView());
```

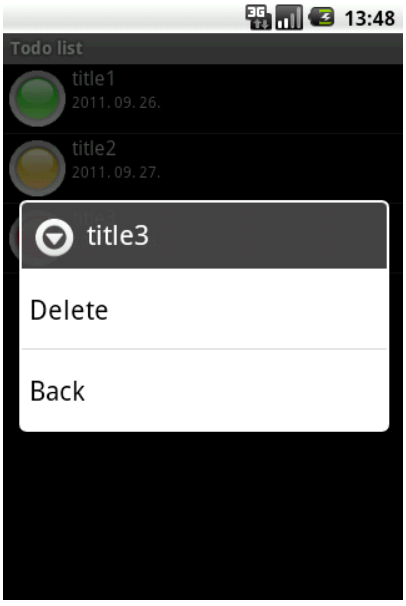
Definiáljuk felül a `onCreateContextMenu(…)` és az `onContextItemSelected(…)` függvényeket:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenuInfo menuInfo) {
     if (v.equals(getListView())) {
          AdapterView.AdapterContextMenuInfo info =
(AdapterView.AdapterContextMenuInfo)menuInfo;

     menu.setHeaderTitle(((Todo)getListAdapter().getItem(in
fo.position)).getTitle());
          String[] menuItems =
getResources().getStringArray(R.array.todomenu);
          for (int i = 0; i<menuItems.length; i++) {
               menu.add(Menu.NONE, i, i, menuItems[i]);
          }
     }
}

@Override
public boolean onContextItemSelected(MenuItem item) {
```

```
     AdapterView.AdapterContextMenuInfo info =
(AdapterView.AdapterContextMenuInfo)item.getMenuInfo();
     int menuItemIndex = item.getItemId();
     if (menuItemIndex==0)
     {

     ((TodoAdapter)getListAdapter()).deleteRow((Todo)getLis
tAdapter().getItem(info.position));

     ((TodoAdapter)getListAdapter()).notifyDataSetChanged()
;
     }
     return true;
}
```

Az `onCreateContextMenu(…)` függvényben hivatkozunk a `R.array.todomenu` erőforrásra, mely egy string tömböt ír le. Hozzunk létre egy `todomenu.xml` nevű állományt a `res/values` könyvtárba, melynek tartalma legyen:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array
        name="todomenu">
        <item>Delete</item>
        <item>Back</item>
    </string-array>
</resources>
```

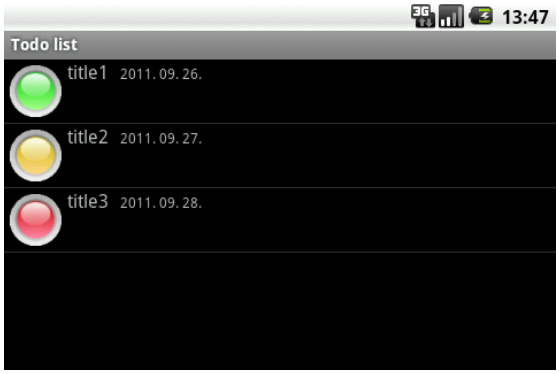


Todo elem törlése

3.3 Landscape orientáció támogatása

A Landscape nézet támogatásában a listaelemek megjelenítését változtassuk meg úgy, hogy a Todo címe és esedékességi dátuma egymás mellett jelenjenek meg és ne egymás alatt. Ehhez hozzunk létre egy layout-land könyvtárat és benne egy új todorow.xml-t, melynek tartalma a következő. Próbáljuk ki emulátoron, a működést!

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="horizontal"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content">

    <ImageView
     android:id="@+id/imageViewPriority"
     android:layout_height="wrap_content"
     android:layout_width="wrap_content"
     android:src="@drawable/high"
     android:padding="5dp"/>

    <LinearLayout
     android:layout_height="wrap_content"
     android:layout_width="fill_parent"
     android:orientation="horizontal">

         <TextView
          android:id="@+id/textViewTitle"
          android:textSize="16dp"
          android:text="Title"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"/>

         <TextView
          android:id="@+id/textViewDueDate"
          android:textSize="12dp"
          android:text="DueDate"
          android:paddingLeft="10dp"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"/>
     </LinearLayout>

</LinearLayout>
```



Landscape mód

3.4 Új Todo elem létrehozása

A kezdő `ListActivity`-hez adjunk egy saját menüt rendelkezik, melyben egy „Create new Todo” menüpont található, melyet kiválasztva dialógus formában egy új `Activity` jelenik meg.

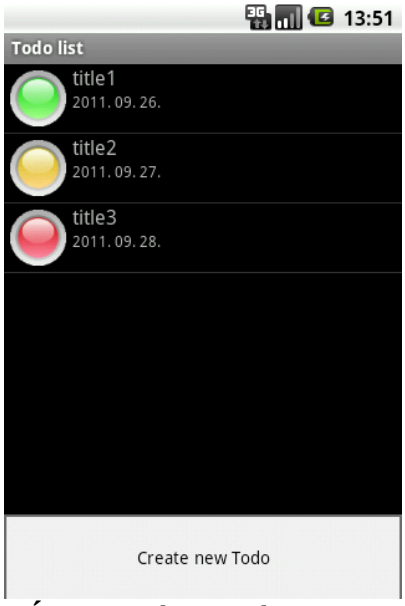
3.4.1 Todo létrehozása menü

Ehhez elsőként definiáljuk felül a `onCreateOptionsMenu`(…) és `onOptionsItemSelected`(…) függvényeket a ListActivity-ben:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.listmenu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

     if (item.getItemId() == R.id.itemCreateTodo)
     {
          Intent myIntent = new Intent();
          myIntent.setClassName(
           "hu.bute.daai.amorg.examples",
          "hu.bute.daai.amorg.examples.ActivityCreateTodo");
          startActivity(myIntent);
     }

     return super.onOptionsItemSelected(item);
}
```

A menü létrehozásakor hivatkozunk az `R.menu.listmenu` erőforrásra, melynek tartalma:

```
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/itemCreateTodo"
          android:title="@string/itemCreateTodo"/>
</menu>
```



Új Todo létrehozása menü

3.4.2 Todo létrehozó Activity

Hozzunk létre egy új `ActivityCreateTodo` `Activity`-t, mely egy `TableLayout`-ban tartalmazza az új *Todo* létrehozásához szükséges elemeket.

A `Manifest` állományba vegyük fel az új `Activity`-t:

```
<activity android:name=".ActivityCreateTodo"
          android:label="@string/itemCreateTodo"
          android:theme="@android:style/Theme.Dialog">
</activity>
```

Az új `ActivityCreateTodo` forrása alább látható. Figyeljük meg, hogy működik a dátumkiválasztó dialógus. A dialógus feldobásához meghívjuk az `Activity` `showDialog(ID)` függvényét, melyben átadunk egy egyedi dialógus azonosítót. Az `Activity` ezután az `onCreateDialog(…)` függvényt hívja meg, melyben

megadjuk, hogy az általunk megadott ID-hez milyen dialógust hozzon létre. Végül pedig az `mDateSetListener` objektum megadásával leírjuk, hogy mi történjen a dátum kiválasztásakor.

```
public class ActivityCreateTodo extends Activity {

     private static final int DATE_DUE_DIALOG_ID = 1;

     private Calendar calSelectedDate =
Calendar.getInstance();

     private EditText editTodoTitle;
     private Spinner spnrTodoPriority;
     private TextView txtDueDate;
     private EditText editTodoDescription;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.createtodo);

        // aktuális dátum beállítása
        calSelectedDate.setTime(new
Date(System.currentTimeMillis()));

        // UI elem referenciák elkérése
        editTodoTitle =
(EditText)this.findViewById(R.id.todoTitle);

        spnrTodoPriority =
(Spinner)this.findViewById(R.id.todoPriority);
        String[] priorities=new String[3];
        priorities[0]="Low";
        priorities[1]="Medium";
        priorities[2]="High";
        spnrTodoPriority.setAdapter(new
ArrayAdapter(this,android.R.layout.simple_spinner_item,
priorities));

        txtDueDate =
(TextView)this.findViewById(R.id.todoDueDate);
        refreshDateText();
        txtDueDate.setOnClickListener(new OnClickListener()
{
                public void onClick(View v) {
```

```
                    showDialog(DATE_DUE_DIALOG_ID);
                }
            });

        editTodoDescription =
(EditText)this.findViewById(R.id.todoDescription);

        // gomb eseménykezelők beállítása
        Button btnOk =
(Button)findViewById(R.id.btnCreateTodo);
        btnOk.setOnClickListener(new OnClickListener() {
                public void onClick(View v) {
                    finish();
                }
            });

        Button btnCancel =
(Button)findViewById(R.id.btnCancelCreateTodo);
        btnCancel.setOnClickListener(new OnClickListener()
{
                public void onClick(View v) {
                    finish();
                }
            });
    }

    private void refreshDateText()
    {
     StringBuilder dateString = new StringBuilder();
     dateString.append(calSelectedDate.get(Calendar.YEAR));
     dateString.append(". ");

     dateString.append(calSelectedDate.get(Calendar.MONTH)+
1);
     dateString.append(". ");

     dateString.append(calSelectedDate.get(Calendar.DAY_OF_
MONTH));
     dateString.append(".");

     txtDueDate.setText(dateString.toString());
    }

    @Override
    protected Dialog onCreateDialog(int id)
    {
```

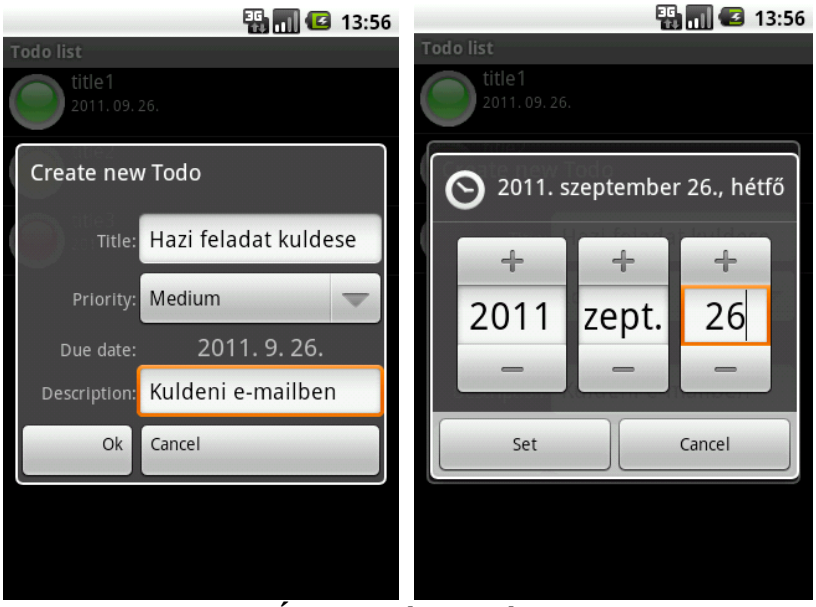
```
        switch (id) {
            case DATE_DUE_DIALOG_ID:
            {
                return new DatePickerDialog(
                    this, mDateSetListener,
calSelectedDate.get(Calendar.YEAR),
                    calSelectedDate.get(Calendar.MONTH),
calSelectedDate.get(Calendar.DAY_OF_MONTH));
            }
        }
        return null;
    }

    private DatePickerDialog.OnDateSetListener
mDateSetListener =
        new DatePickerDialog.OnDateSetListener()
        {
            public void onDateSet(DatePicker view, int
year, int monthOfYear,
                    int dayOfMonth)
            {
               // Új dátum beállítása
               calSelectedDate.set(Calendar.YEAR, year);
               calSelectedDate.set(Calendar.MONTH,
monthOfYear);
               calSelectedDate.set(Calendar.DAY_OF_MONTH,
dayOfMonth);
               refreshDateText();
                removeDialog(DATE_DUE_DIALOG_ID);
            }
        };
}
```

Az ActivityCreateTodo layout-ját a createtodo.xml-ben adjuk meg:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:layout_column="1"
            android:text="@string/lblTodoTitle"
            android:layout_width="wrap_content"
```

```
            android:gravity="right"/>
        <EditText
          android:id="@+id/todoTitle"
            android:width="200dp"/>
    </TableRow>
    <TableRow>
        <TextView
            android:layout_column="1"
            android:text="@string/lblTodoPriority"
            android:layout_width="wrap_content"
            android:gravity="right"/>
        <Spinner
          android:id="@+id/todoPriority"
            android:width="200dp"/>
    </TableRow>
    <TableRow>
        <TextView
            android:layout_column="1"
            android:text="@string/lblTodoDueDate"
            android:layout_width="wrap_content"
            android:gravity="right"/>
        <TextView
            android:id="@+id/todoDueDate"
            android:textSize="20dp"
            android:width="200dp"
            android:gravity="center"/>
    </TableRow>
    <TableRow>
        <TextView
            android:layout_column="1"
            android:text="@string/lblTodoDescription"
            android:layout_width="wrap_content"
            android:gravity="right"/>
        <EditText
          android:id="@+id/todoDescription"
            android:width="200dp"/>
    </TableRow>

     <TableRow>
        <Button
          android:id="@+id/btnCreateTodo"
            android:layout_column="1"
            android:text="@string/btnOk"
            android:layout_width="wrap_content"
            android:gravity="right"/>
        <Button
```

```
            android:id="@+id/btnCancelCreateTodo"
              android:text="@string/btnCancel"
              android:layout_width="wrap_content"
              android:gravity="left"/>
    </TableRow>
</TableLayout>
```



Új Todo létrehozása

3.4.3 Todo elmentése

A *Todo* elmentéséhez ebben a példában egy egyszerűsített, ronda megoldást fogunk választani, mivel a kívánatos megoldást majd a 4. (*Intent*) előadás után tudjuk megmutatni. A jelenlegi egyszerűsített megoldásban létrehozunk egy `DataPreferences` osztályt, mely tartalmaz egy `public static Todo todoToCreate = null;` objektumot. A *Todo* létrehozó `Activity` *OK* gombjának választásakor ebben az objektumban mentjük el a létrehozandó *Todo*-t és az eredeit `ListActivity onResume()` függvényét felüldefiniálva megvizsgáljuk ennek a `todoToCreate` objektumnak a nullitását. Amennyiben nem *null*, felveszünk egy új *Todo* elemet és *null*-ra állítjuk a `todoToCreate`-t. A *Todo* létrehozó `Activity` *Cancel* gombjára szintén *null*-ra állítjuk a `todoToCreate` objektumot.

A `DataPreferences` osztály:

```
public class DataPreferences {
     public static Todo todoToCreate = null;
}
```

Az *OK* és a *Cancel* gombok eseménykezelői a *Todo*-t létrehozó `Activity`-ben:

```
// gomb eseménykezelők beállítása
Button btnOk = (Button)findViewById(R.id.btnCreateTodo);
btnOk.setOnClickListener(new OnClickListener() {
     public void onClick(View v) {
          Todo.Priority selectedPriority =
Todo.Priority.LOW;

          switch
(spnrTodoPriority.getSelectedItemPosition()) {
               case 0:
                    selectedPriority = Todo.Priority.LOW;
                    break;
               case 1:
                    selectedPriority =
Todo.Priority.MEDIUM;
                    break;
               case 2:
                    selectedPriority = Todo.Priority.HIGH;
                    break;
               default:
                    break;
          }

          // NAGYON RONDA MEGOLDÁS
          // Intent előadáson megmutatjuk a helyeset!
          DataPreferences.todoToCreate = new Todo(
                    editTodoTitle.getText().toString(),
                    selectedPriority,
                    txtDueDate.getText().toString(),

     editTodoDescription.getText().toString()
          );

          finish();
     }
});

Button btnCancel =
(Button)findViewById(R.id.btnCancelCreateTodo);
btnCancel.setOnClickListener(new OnClickListener() {
     public void onClick(View v) {
          DataPreferences.todoToCreate = null;
          finish();
     }
});
```

Végül *Todo*-k listáját tartalmazó `ListActivity` `onResume(…)` függvénye:

```
@Override
protected void onResume() {
      super.onResume();

      // NAGYON RONDA MEGOLDÁS
      // Intent előadáson megmutatjuk a helyeset!
      if (DataPreferences.todoToCreate != null)
      {
      ((TodoAdapter)getListAdapter()).addItem(DataPreferences.tod
oToCreate);
            DataPreferences.todoToCreate = null;

      ((TodoAdapter)getListAdapter()).notifyDataSetChanged();
      }
}
```

Todo elmentése

Megjegyzés:
Mivel a Todo elemeket nem mentjük el perzisztensen adatbázisba, ezért a képernyő elforgatásakor újra létrejön az Activity és emiatt törlődnek az eddig létrehozott Todo elemeink. A probléma helyes kezelésére egy későbbi előadásban és laboron mutatunk eszközöket (SQLite adatbázis használata).