BUTE Department of Automation and Applied Informatics
AMORG
APPLIED MOBILE RESEARCH GROUP

Andriod alapú szoftverfejlesztés

3. Labor

Alapvető alkalmazás készítése Fragment-ek használatával

Tartalom

1 Felkészülés a mérésre

A mérés célja az Android Fragment keretrendszer és a *Support library* megismerése. A mérés során a korábban elkészített Todo alkalmazást alakítjuk át. A cél a már korábban implementált funkciók Fragment-ekbe való szervezése. Ennek során az alábbi témákat fogjuk érinteni:

- `Fragment` és `ListFragment` osztályok
- `FragmentActivity` használata a *Support library*-n keresztül
- Fragment-tranzakciók indítása a `FragmentManager` osztályon keresztül
- Kommunikáció a Fragment-ek közt
- Több különböző képernyőméret támogatása

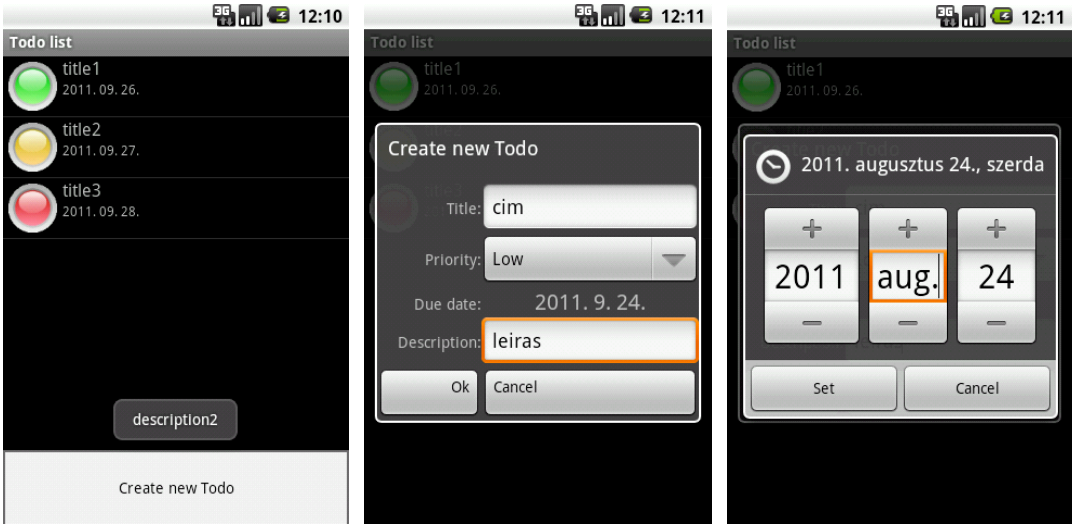
2 Laborfeladatok

A legutóbbi labor során egy Todo lista alkalmazást hoztunk 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 mostani 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

2.1 A projekt és az ActivityMain osztály létrehozása

Első lépésként hozzuk létre a projektet egy új névvel (pl. TodoBasicFragment). Build SDK szintnek 4.1-et, Minimum Required SDK szintnek pedig 2.2-t válasszunk. A varázslótól rögtön kérjük is egy kezdő BlankActivity létrehozását is, `MainActivity` névvel.

2.2 Újrahasznosítás

Az alábbi - a korábbi mérés során már létrehozott - erőforrásokat és osztályokat emeljük át az új projektbe:

- Todo osztály
- TodoAdapter osztály
- A low/medium/high.png kép-erőforrások
- A /res/values/string.xml string-erőforrások
- A /res/values/todomenu.xml string-erőforrások
- A /res/layout/todorow.xml erőforrás
- A /res/layout/createtodo.xml erőforrás
- A /res/menu/listmenu.xml erőforrás

2.3 TodoListFragment létrehozása

Hozzunk létre egy `TodoListFragment` nevű osztályt, amely a `ListFragment`-ből származik. A `ListFragment` egy a `ListActivity`-hez hasonló segédosztály, melyet akkor érdemes használni, ha a Fragment feladata elsősorban egy `ListView` megjelenítése és kezelése. Használata során nem kell az `onCreateView()` metódust felülírnunk a `ListView` létrehozásához, mivel ezt már az ősosztály megteszi nekünk. Referenciát erre az automatikusan létrehozott listára a `getListView()` metódussal kapunk. Emellett az `onListItemClick()` metódus felüldefiniálásával azt is megadhatjuk, hogy mit történjen, ha a felhasználó kiválaszt egy listaelemet.

Az `onCreate()` metódusban jelezzük a `setHasOptionMenu(true)` hívással, hogy a Fragment rendelkezik OptionsMenu-vel:

```
@Override
public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);

	setHasOptionsMenu(true);
}
```

Az `onStart()` metódusban hozzuk létre a listához szükséges adapter, töltsük fel néhány elemmel, majd rendeljük hozzá a listához. Emellett jelezzük a keretrendszernek, hogy a listához Context menü is tartozik.

```
@Override
public void onStart() {
	super.onStart();

	// Adapter letrehozasa esfeltoltese nehany 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.",	"description3"));
		adapter = new TodoAdapter(getActivity(), todos);
		setListAdapter(adapter);

	registerForContextMenu(getListView());
```

```
}
```

Adjuk meg a Context menühöz kapcsolódó callback metódusokat is, a korábbi laborhoz hasonlóan:

```
@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(info.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) getListAdapter()
                         .getItem(info.position));
          ((TodoAdapter)
getListAdapter()).notifyDataSetChanged();
     }
     return true;
}
```

Az `onListItemClick()` metódus felüldefiniálásával adjuk meg, hogy mi történjék, ha a felhasználó kiválaszt egy *Todo* elemet. Ekkor a már az előadáson bemutatott módon, egy Listener interfészen keresztül értesíti a Fragment az erre az eseményre feliratkozott objektumokat:

```
@Override
public void onListItemClick(ListView l, View v, int
     position,      long id) {
     super.onListItemClick(l, v, position, id);
```

```
        Todo selectedTodo = (Todo)
        getListAdapter().getItem(position);

        if (listener != null) {
            listener.onTodoSelected(selectedTodo);
        }
}
```

Definiáljuk a Listener interfészt is, melyet a Fragment-hez tartozó szülő Activity-nek implementálnia kell, ha értesülni szeretne egy-egy *Todo* elem kiválasztásáról:

```
public interface ITodoListFragment {
    public void onTodoSelected(Todo selecedTodo);
}
```

A Fragment `onAttach()` metódusában regisztráljuk be (legalábbis próbáljuk meg) a szülő Activity-t, mint Listener objektumot:

```
@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);

    try {
        listener = (ITodoListFragment) activity;
    } catch (ClassCastException ce) {
        Log.e(TAG, "Parent Activity does not implement
    listener interface!");
    } catch (Exception e) {
        Log.e(TAG, "Unexpected exception!");
        e.printStackTrace();
    }
}
```

Dekleráljuk a fenti kódrészletekhez szükséges mezőket is az osztályban:

```
// Log tag
public static final String TAG = "TodoListFragment";

// State
private TodoAdapter adapter;

// Listener
private ITodoListFragment listener;
```

2.4 A TodoListFragment használata

A `TodoListFragment`-et a `MainActivity` fogja statikus módon felhasználni. Ehhez az activity_main.xml erőforrás-fájlt az alábbi módon definiáljuk:

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

    <fragment
     class="hu.bme.todo.todobasicfragmentlabor.fragment.Tod
oList          Fragment"
        android:tag="TodoListFragment"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"/>

</LinearLayout>
```

Módosítsuk a `MainActivity` osztály, hogy az Activity helyett a `FragmentActivity`-ből származzon. A `FragmentAcitivty` osztályt a *Support library* tartalmazza, ez teszi lehetővé a Fragment keretrendszer használatát a korábbi Android rendszereken.

```
public class MainActivity extends FragmentActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

}
```

Próbáljuk ki az alkalmazást!

3 A TodoDetailsFragment létrehozása

Az alkalmazás új verziója a kiválasztott *Todo* elem Description tulajdonságát már nem Toast üzenetben, hanem egy annak dedikált Fragment-ben fogja megjeleníteni. Ehhez készítsük el a `TodoDetailsFragment` osztályt, amely a `Fragment`-ből származik.

3.1 Factory metódusok

A `TodoDetailsFragment` létrehozásához paramétereket is át kell adnunk (a megjelenítendő Todo elem tulajdonságait). Készítsünk statikus factory metódusokat, amelyek elvégzik ezt a munkát és megkönnyítik az osztály helyes használatát:

```
public static TodoDetailsFragment newInstance(String
todoDesc) {
     TodoDetailsFragment result = new
TodoDetailsFragment();        Bundle args = new Bundle();
     args.putString(KEY_TODO_DESCRIPTION, todoDesc);
     result.setArguments(args);

     return result;
}

public static TodoDetailsFragment newInstance(Bundle args)
{
     TodoDetailsFragment result = new
TodoDetailsFragment();

     result.setArguments(args);

     return result;
}
```

Definiáljuk az osztály működéséhez szükséges mezőket is:

```
public static final String TAG = "TodoDetailsFragment";

public static final String KEY_TODO_DESCRIPTION =
     "todoDesc";

private TextView todoDescription;

private static Todo selectedTodo;
```

3.2 Fragment életciklus

Az `onCreate()` metódusban a megkapott paraméterek alapján beállítjuk a *selectedTodo* tulajdonságot:

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      if (savedInstanceState == null) {
            if (getArguments() != null) {
                  selectedTodo = new Todo("cim",
      Priority.LOW,  "1987.23.12",

      getArguments().getString(KEY_TODO_DESCRIPTION));
            }
      }
}
```

Az `onCreateView()` hívásban fel kell építenünk a Fragment-hez tartozó UI-t:

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup
      container,     Bundle savedInstanceState) {
            View root =
      inflater.inflate(R.layout.fragment_todo_details,
container,
                        false);

      todoDescription = (TextView)
      root.findViewById(R.id.todoDescription);

      todoDescription.setText(selectedTodo.getDescription())
;

      return root;
}
```

Végül definiáljuk a Fragment-hez tartozó layout XML fájlt is (fragment_todo_details.xml):

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

    <TextView
        android:id="@+id/todoDescription"
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

3.3 A TodoDetailsFragment használata

Készen áll a *Todo* adatait tartalmazó Fragment is, most már csak helyesen kell használni. Használjuk az előadáson bemutatott módszer, fedjük le egyszerre a mobil és tablet eszközökön való megjelenítést.

Módosítsuk a fő Activity-nket, úgy hogy megfeleljen a `TodoListFragment`-ben definiált interfésznek. Vegyünk fel egy `ViewGroup`-ot, és implementáljuk az `ITodoListFragment` interfészt. Írjuk meg a szükséges metódust. Így az Activity tud arról, ha egy listaelemre kattintanak, mert a `TodoListFragment` meghívja a listener-ként beregisztrált Activity `onTodoSelected()` metódusát.

```
public class ActivityMain extends FragmentActivity implements
ITodoListFragment {

      private ViewGroup fragmentContainer;

      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            setContentView(R.layout.activity_main);

            fragmentContainer = (ViewGroup)
findViewById(R.id.FragmentContainer);
      }

      // ITodoListFragment
      public void onTodoSelected(Todo selectedTodo) {
            if (fragmentContainer != null) {
                  FragmentManager fm = getSupportFragmentManager();

                  FragmentTransaction ft = fm.beginTransaction();
                  ft.replace(R.id.FragmentContainer,
TodoDetailsFragment

      .newInstance(selectedTodo.getDescription()));
                  ft.commit();
            } else {
                  Intent i = new Intent(this,
ActivityTodoDetails.class);
```

```
        i.putExtra(TodoDetailsFragment.KEY_TODO_DESCRIPTION,
                                selectedTodo.getDescription());
                    startActivity(i);
                }
        }
}
```

A fenti kódrész helyes működéséhez módosítsuk a nagy képernyős eszközökhöz tartozó layout xml fájlt (res/layout-large/activity_main.xml). Nagy képernyőméret mellett az Activity tartalmaz egy `FrameLayout`-ot is *FragmentContainer* ID-vel, amely képes fogadni a kiválasztott Todo elem részleteit megjelenítő Fragment-eket:

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

    <fragment
class="hu.bute.daai.amorg.examples.fragment.TodoListFragmen
t"
        android:tag="TodoListFragment"
        android:layout_width="0dip"
        android:layout_height="fill_parent"
        android:layout_weight="1"/>

    <FrameLayout
        android:id="@+id/FragmentContainer"
        android:layout_width="0dip"
        android:layout_height="fill_parent"
        android:layout_weight="2"/>

</LinearLayout>
```

Kisebb képernyőméret esetén az alkalmazás az alapértelmezett /res/layout mappából tölti be a már korábban definiált activity_main.xml-t, amely nem tartalmazza a *FragmentContainer* elemet. Ilyen esetben egy külön Activity-ben jelenítjük meg a kiválasztott *Todo* elem részleteit megjelenítő Fragment-et.

3.3.1 DetailsActivity a mobilokra

Készítsünk egy `FragmentActivity`-ből származó `ActivityTodoDetails` nevű osztályt.

```
public class ActivityTodoDetails extends FragmentActivity {

     @Override
     protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);

          setContentView(R.layout.activity_details);

          if (savedInstanceState == null &&
getIntent().getExtras() != null) {
               Bundle args = new
Bundle(getIntent().getExtras());
               TodoDetailsFragment detailsFragment =
TodoDetailsFragment
                         .newInstance(args);

               // Add details fragment
               FragmentManager fm =
getSupportFragmentManager();

               FragmentTransaction ft =
fm.beginTransaction();
               ft.add(R.id.FragmentContainer,
detailsFragment);
               ft.commit();
          }
     }
}
```

(!Ne felejtsük el felvenni a Manifestben az új Activity-t!)

Definiáljuk az `ActivityTodoDetails`-hoz szükséges layout xml fájlt is:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/FragmentContainer"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" />
```

Ellenőrizzük a működést mind tablet, mind telefon emulátoron.

Nem kötelező, de ajánlott:
Az Intentekről még mindig nem volt előadás, azonban nem nehéz kikövetkeztetni a kód működését. A savedInstanceState jelentőségét már ismerjük. Gondolkodjunk el egy kicsit

az Intent putExtra metódusán és a részleket tartalmazó Activity getIntent majd annak getExtras metódusán. Kérdezzünk a laborvezetőktől.

4 Todo létrehozása

4.1 Új Todo elem létrehozása

A kezdő `TodoListFragment`-hez adjunk egy saját menüt, melyben egy „Create new Todo” menüpont található, melyet kiválasztva dialógus formában egy új `DialogFragment` jelenik meg, hasonlóan a korábbi laboron látott megoldáshoz.

4.1.1 Todo létrehozása menü

Ehhez elsőként definiáljuk felül a `onCreateOptionsMenu(…)` és `onOptionsItemSelected(…)` metódusokat a TodoListFragment-ben:

```
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater
inflater) {
     inflater.inflate(R.menu.listmenu, menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

     if (item.getItemId() == R.id.itemCreateTodo) {
          TodoCreateFragment createFragment = new
TodoCreateFragment();
          createFragment.setTargetFragment(this, 0);

          FragmentManager fm = getFragmentManager();
          createFragment.show(fm, TodoCreateFragment.TAG);
     }

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

Az új Fragment-nek a `.setTargetFragment()` metódussal beállítjuk magunkat (`TodoListFragment`), amit majd lekérdez a `getTargetFragment`-tel a `TodoCreateFragment`. Így tudja meg, hogy ki a listener, akit értesítenie kell egy új Todo elem létrejöttekor.

4.1.2 Todo létrehozó Fragment

Készítsünk egy új osztályt `TodoCreateFragment` néven ami a `DialogFragment`-ből származik.

A kód nagyon hasonlít a legutóbbi laboron megismert Activity alapú megoldásra. Az `onCreateView`-ban történik mindaz, ami a múltkor az `onCreate`-ben volt. Az `onAttach` hívás során ellenőrizzük, hogy van-e listener objektum beregisztrálva a dialógusunk számára. A `TodoListFragment` fog értesülni az új Todo-ról, úgy ahogyan a `CreateTodoFragment`-ünk is értesülni fog a dátumválasztásról.

```
public class TodoCreateFragment extends DialogFragment {

	// Log tag
	public static final String TAG = "TodoCreateFragment";

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

	// Listener
	private ITodoCreateFragment listener;

	@Override
	public void onAttach(Activity activity) {
		super.onAttach(activity);

		if (getTargetFragment() != null) {
			try {
				listener = (ITodoCreateFragment)
getTargetFragment();
			} catch (ClassCastException ce) {
				Log.e(TAG,
					"Target Fragment does not
implement fragment interface!");
			} catch (Exception e) {
				Log.e(TAG, "Unhandled exception!");
				e.printStackTrace();
			}
		} else {
			try {
				listener = (ITodoCreateFragment)
activity;
			} catch (ClassCastException ce) {
```

```
                    Log.e(TAG,
                              "Parent Activity does not
implement fragment interface!");
               } catch (Exception e) {
                    Log.e(TAG, "Unhandled exception!");
                    e.printStackTrace();
               }
          }
     }

     @Override
     public View onCreateView(LayoutInflater inflater,
ViewGroup container,
               Bundle savedInstanceState) {
          View root = inflater.inflate(R.layout.createtodo,
container, false);

          // Dialog cimenek beallitasa
          getDialog().setTitle(R.string.itemCreateTodo);

          // UI elem referenciak elkerese
          editTodoTitle = (EditText)
root.findViewById(R.id.todoTitle);

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

          txtDueDate = (TextView)
root.findViewById(R.id.todoDueDate);
          txtDueDate.setText("  -  ");
          txtDueDate.setOnClickListener(new
OnClickListener() {
               public void onClick(View v) {
                    //Itt jon a datumvalaszto
               }
          });

          editTodoDescription = (EditText) root
```

```
                    .findViewById(R.id.todoDescription);

          // A gombok esemenykezeloinek beallitasa
          Button btnOk = (Button)
root.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;
                    }

                    if (listener != null) {
                         listener.onTodoCreated(new
Todo(editTodoTitle.getText()
                                   .toString(),
selectedPriority, txtDueDate.getText()
                                   .toString(),
editTodoDescription.getText()
                                   .toString()));
                    }

                    dismiss();
               }
          });

          Button btnCancel = (Button)
root.findViewById(R.id.btnCancelCreateTodo);
          btnCancel.setOnClickListener(new
OnClickListener() {
```

```
                public void onClick(View v) {
                    dismiss();
                }
            });

            return root;
        }


        // Listener interface
        public interface ITodoCreateFragment {
            public void onTodoCreated(Todo newTodo);
        }
}
```

Most ugorjunk vissza a `TodoListFragment`-re. Valósítsuk meg az `ITodoCreateFragment` interfészt és írjuk meg a szükséges metódust.

```
public class TodoListFragment extends ListFragment
implements
        ITodoCreateFragment {
...


        // ITodoCreateFragment
        public void onTodoCreated(Todo newTodo) {
            adapter.addItem(newTodo);
            adapter.notifyDataSetChanged();
        }
}
```

Ezek után ellenőrizzük, hogy működik az új *Todo* felvitele (kivéve a dátumválasztást).

4.1.3 Dátumválasztás

A dátumválasztás módszere immár gyerekjáték. A `CreateTodoFragment`-ünk implementálja a `IDatePickerDialogFragment` interfészét a `DatePickerDialogFragment`-ünknek, így a dátumválasztásról értesül az új Todo felvitele `DialogFragment`-ünk.
Először is csináljuk még egy `DialogFragment`-ből származó osztályt, ezúttal nevezzük `DatePickerDialogFragment`-nek.

```
public class DatePickerDialogFragment extends
DialogFragment {
```

```
	// Log tag
	public static final String TAG =
"DatePickerDialogFragment";

	// State
	private Calendar calSelectedDate =
Calendar.getInstance();

	// Listener
	private IDatePickerDialogFragment listener;

	@Override
	public void onAttach(Activity activity) {
		super.onAttach(activity);

		if (getTargetFragment() != null) {
			try {
				listener = (IDatePickerDialogFragment)
getTargetFragment();
			} catch (ClassCastException ce) {
				Log.e(TAG,
					"Target Fragment does not
implement fragment interface!");
			} catch (Exception e) {
				Log.e(TAG, "Unhandled exception!");
				e.printStackTrace();
			}
		} else {
			try {
				listener = (IDatePickerDialogFragment)
activity;
			} catch (ClassCastException ce) {
				Log.e(TAG,
					"Parent Activity does not
implement fragment interface!");
			} catch (Exception e) {
				Log.e(TAG, "Unhandled exception!");
				e.printStackTrace();
			}
		}
	}

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
```

```
            calSelectedDate.setTime(new
Date(System.currentTimeMillis()));
      }

      @Override
      public Dialog onCreateDialog(Bundle
savedInstanceState) {
            return new DatePickerDialog(getActivity(),
mDateSetListener,
                        calSelectedDate.get(Calendar.YEAR),
                        calSelectedDate.get(Calendar.MONTH),

      calSelectedDate.get(Calendar.DAY_OF_MONTH));
      }

      private DatePickerDialog.OnDateSetListener
mDateSetListener = new DatePickerDialog.OnDateSetListener()
{
            public void onDateSet(DatePicker view, int year,
int monthOfYear,
                        int dayOfMonth) {
                  // uj datum beallitasa
                  calSelectedDate.set(Calendar.YEAR, year);
                  calSelectedDate.set(Calendar.MONTH,
monthOfYear);
                  calSelectedDate.set(Calendar.DAY_OF_MONTH,
dayOfMonth);

                  if (listener != null) {

      listener.onDateSelected(buildDateText());
                  }

                  dismiss();
            }
      };

      private String buildDateText() {
            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(".");

            return dateString.toString();
        }

        public interface IDatePickerDialogFragment {
            public void onDateSelected(String date);
        }
}
```

Ugorjunk vissza a `CreateTodoFragment`-re és valósítsuk meg az `IDatePickerDialog` interfészt, illetve állítsuk be a `txtDueDate` `onClickListener`-jében, hogy mutassunk egy DialogFragmen-tet.

Itt látszik a `DialogFragment` előnye az `Activity`-s megoldáshoz képest: a dátumválasztó dialógusunk nem üti ki a Todo készítő dialógusunkat. Az új Todo dialógus és a dátumválasztó dialógus kommunikációja a `set`- és `getTargetFragment()`-en keresztül működik. Az indító fél átadja önmagát, aki ellátja a listener funkciót. Az új Fragment a `getTargetFragment()`-en keresztül elkéri az indítót, és értesíti a változásról.

```
public class TodoCreateFragment extends DialogFragment
implements
            IDatePickerDialogFragment {
…
        txtDueDate.setOnClickListener(new OnClickListener() {
                    public void onClick(View v) {
                        showDatePickerDialog();
                    }
            });
…


        private void showDatePickerDialog() {
            FragmentManager fm = getFragmentManager();

            DatePickerDialogFragment datePicker = new
DatePickerDialogFragment();
            datePicker.setTargetFragment(this, 0);
            datePicker.show(fm,
DatePickerDialogFragment.TAG);
```

```
    }

    // IDatePickerDialogFragment
    public void onDateSelected(String date) {
        txtDueDate.setText(date);
    }
```