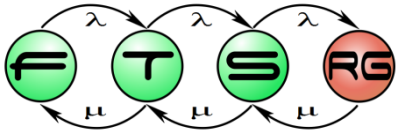
Magas szintű GUI programozás

JFace



SWT és JFace

- SWT
  - Natív
  - Alacsony szintű elemkészlet
  - Jól kézben tartható működés
  - Sok kódolás
- JFace
  - Magas szintű komponensek (SWT-re épít)
  - Jobban automatizált
  - Struktúráltabb szerkezet
  - Könnyebb újrafelhasználás
  - Kevésbé kézben tartható

JFace ApplicationWindow

- Alkalmazás ablak kezelő osztály
  - Az ablak megjelenítése: SWT Shell segítségével
  - Támogatás menüsor, eszköztár, státuszsor készítéséhez
- Metódusok
  - createControls(Composite parent)
    - Form tartalmának létrehozása
  - addCoolbar(int style) vagy addToolbar(int style)
    - Eszköztárak létrehozása
  - addStatusLine()
    - Státuszsor létrehozása
  - createMenuManager()

JFace Hello World

```
public class MyApp extends ApplicationWindow {
   public MyApp()  {
       super(null);
   }
   @Override
   protected Control createContents(Composite parent) {
       Text helloText = new Text(parent, SWT.CENTER);
       helloText.setText("Hello JFace world!");
       parent.pack();
       return parent;
   }
   public static void main(String[] args){
       MyApp p = new MyApp();
       p.setBlockOnOpen(true);
       p.open();
       Display.getCurrent().dispose();
```

JFace Hello World

```
public class MyApp extends ApplicationWindow {
   public MyApp()  {
       super(null);
   }
   @Override
   protected Control createCont                    {
       Text helloText = new Text(parent, SWT.CENTER);
       helloText.setText("Hello JFace world!");
       parent.pack();
       return parent;
   }
   public static void main(String[] args){
       MyApp p = new MyApp();
       p.setBlockOnOpen(true);
       p.open();
       Display.getCurrent().dispose();
```

Alkalmazásablak objektum
Menü, eszköztár, ... kezelés

JFace Hello World

```
public class MyApp extends ApplicationWindow {
   public MyApp()  {
       super(null);
   }
   @Override
   protected Control createContents(Composite parent) {
       Text helloText = new Text(parent, SWT.CENTER);
       helloText.setText("Hello, World!");
       parent.pack();
       return parent;
   }
   public static void main(String[] args){
       MyApp p = new MyApp();
       p.setBlockOnOpen(true);
       p.open();
       Display.getCurrent().dispose();
```

Elemek létrehozása

JFace Hello World

```
public class MyApp extends ApplicationWindow {
   public MyApp()  {
       super(null);
   }
   @Override
   protected Control createContents(Composite parent) {
       Text helloText = new Text(parent, SWT.CENTER);
       helloText.setText("Hello JFace world!");
       paren
       retur
   }
   public stat  void main(String[] args){
       MyApp p = new MyApp();
       p.setBlockOnOpen(true);
       p.open();
       Display.getCurrent().dispose();
```

Blokkoló open() hívás

JFace Hello World

```
public class MyApp extends ApplicationWindow {
   public MyApp()  {
       super(null);
   }
   @Override
   protected Control createContents(Composite parent) {
       Text helloText = new Text(parent, SWT.CENTER);
       helloText.setText("Hello JFace world!");
       parent.pack();
       return parent;
   }
   public static void main(Str
       MyApp p = new MyApp();
       p.setBlockOnOpen(true);
       p.open();
       Display.getCurrent().dispose();
```

Mindenki a Display gyereke, elég azt törölni

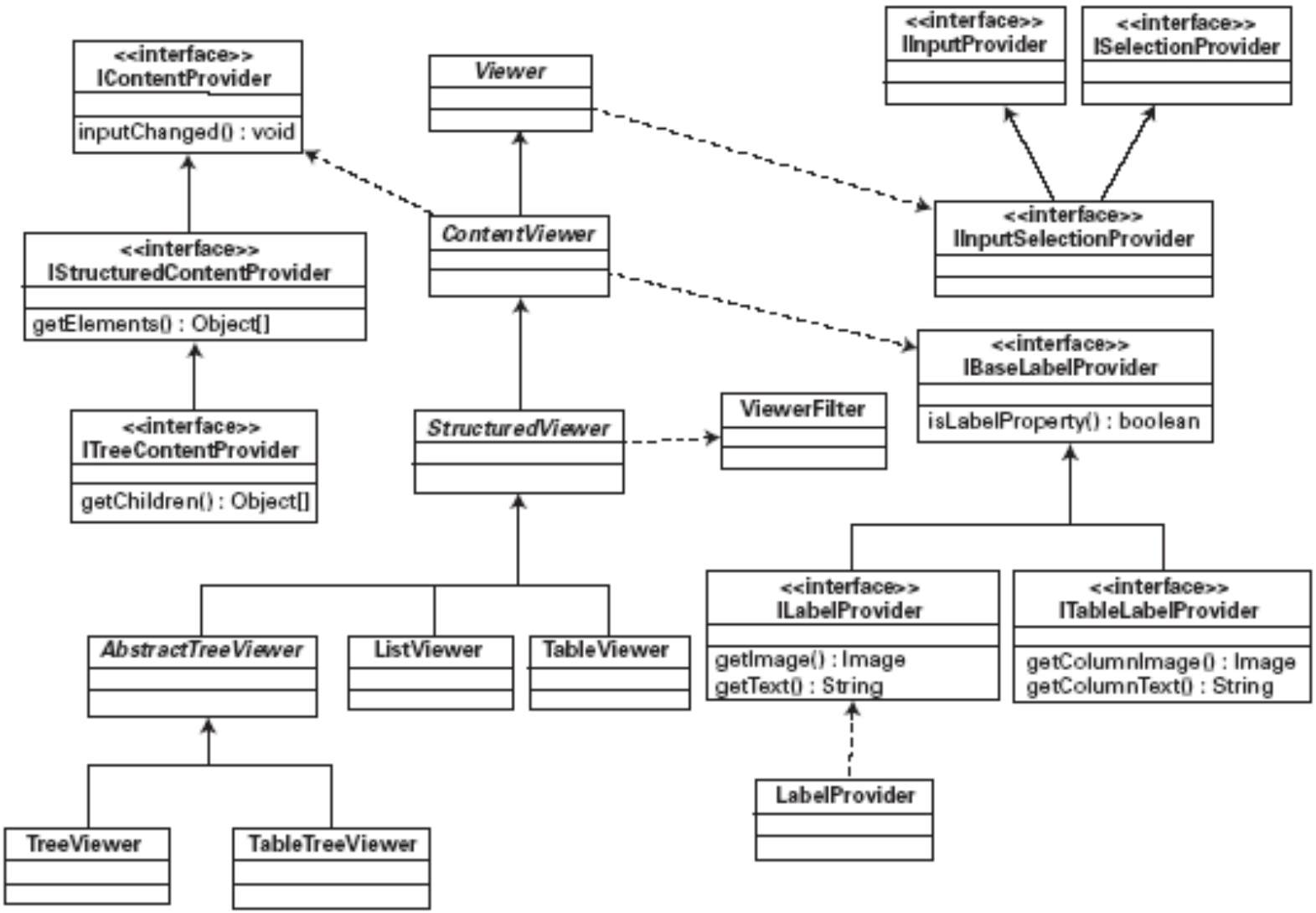
JFace Viewer framework

JFace Viewer framework

- Többféle widget egységes kezelése
  - SWT Table, Tree és List
- MVC minta
  - Modell: ContentProvider, LabelProvider
  - Nézet: Viewer
  - Vezérlő: Listeners

Fontos! Eclipse View és JFace Viewer nem ugyanaz

JFace Viewer framework



Content Provider

- A megjelenítendő elemeket adja meg
- getElements()
  - Elemek tömbjét adja vissza
  - Nem kötelező használni
    - Az elemek a viewer add() metódussal is hozzáadhatóak
- inputChanged(Viewer, Object, Object)
  - Jelzi a providernek, hogy a root objektum megváltozott
    - Ezután a getElements() is hívódni fog
    - Ne hívjuk meg közvetlenül, helyette viewer.setInput(Object)

Label Provider

- Elemekhez felirat/kép rendelése
  - `getText()`
  - `getImage()`
  - `isLabelProperty()`
    - Érinti-e a feliratot a tulajdonság megváltozása?
  - Alapértelmezett megvalósítás
    - Az elemek `toString()` metódusát használja
    - Képet nem ad vissza

Listenerek

- Függ a viewer típusától
- TreeViewer
  - ItemSelection
  - Fa események
- StructuredViewer
  - doubleClick
- getControl() metódus visszaadja az SWT komponenst
  - SWT eseménykezelők elérése

TreeViewer

- ITreeContentProvider
  - A megjelenítendő elemeket adja meg
  - getChildren()
    - Adott elem gyermekeit listázza
  - hasChildren()
    - Vannak-e egy csomópontnak gyerekei
    - Ha lassú kiszámolni, legyen igaz
  - getParent()
    - Szülő visszaadása

ListViewer

- Elemek listájának megjelenítésére
- IStructuredContentProvider
  - getElements()
    - A lista elemeit adja vissza
- Minden egyéb elem használható
  - Sorter
  - Filter
  - Label Provider

Kiválasztás

- getSelection()
  - ISelection
    - Általános jelzése a kijelölésnek
    - Közvetlenül nem használható
  - IStructuredSelection
    - Az elemek sorrendje kötött
    - Iterator-ral bejárható
  - ITreeSelection
    - IStructuredSelection leszármazott
    - Hierarchia leírására

TableViewer

- Táblázat
- TableLayout – a táblázat oszlopainak elrendezése
  - addColumnData()
- A mögötte levő Table elérhető
  - getTable()
- ITableLabelProvider
  - Adott sor és oszlop tartalmának megadására

Táblázatok szerkesztése

- CellEditor
  - ICellModifier
    - getValue()
      - Érték elővétele az objektumból
    - canModify()
      - Szerkeszthető-e az érték
    - Modify()
      - Új érték beírása
  - CellEditor
    - Beépített: Checkbox, Combo box, pop-up dialog, text
    - Lehet sajátot is írni

TableViewer problémák

- Problémák a táblázatok kezelésével
  - Oszlopokat sorszám szerint lehet csak azonosítani
  - Oszloponként egy editort lehet definiálni
    - Problémás a szerkesztett sort figyelembe venni az editornál
      - Pl. legördülő lista tartalma függ a szerkesztett elemtől
  - Sok, könnyen eltéveszthető kód szükséges
- Eclipse 3.3 óta létezik egy újabb API erre

TableViewerColumn

- Egy oszlop definíciója a táblában
- Saját LabelProvider
  - Ajánlott a ColumnLabelProvider osztályból örököltetni
- Saját szerkesztési támogatás
  - Editing Support

EditingSupport

- Kiegészítő osztály szerkesztéshez
- canEdit(Object element)
  - Szerkeszthető-e az adott sorbeli cella
- getCellEditor(Object element)
  - Visszaadja a cellához tartozó CellEditort
- setValue(Object element, Object value)
  - Érték beállítása a modellobjektumban
- getValue(Object element)
  - Visszaadja az elemhez tartozó értéket – ezt a CellEditornak kell feldolgoznia

EditingSupport példa

```
protected abstract class AbstractEditingSupport
  extends
    EditingSupport {

  private TextCellEditor editor;

  public AbstractEditingSupport(TableViewer viewer)
  {
    super(viewer);
    editor = new TextCellEditor(viewer.getTable());
  }

  protected boolean canEdit(Object element) {
    return true;
  }
```

EditingSupport példa

```
protected abstract class AbstractEditingSupport
  extends
    EditingSupport {

  private TextCellEditor editor;

  public AbstractEditin                    r viewer)
  {
    super(viewer);
    editor = new TextCellEditor(viewer.getTable());
  }

  protected boolean canEdit(Object element) {
    return true;
  }
```

Szöveges CellEditor definiálása TableViewerhez

EditingSupport példa

```
protected abstract class AbstractEditingSupport
  extends
    EditingSupport {

  private TextCellEditor editor;

  public AbstractEditingSupport(TableViewer viewer)
  {
    super(viewer);
    editor = new TextCellEditor(viewer.getTable());
  }

  protected boolean canEdit(Object element) {
    return true;
  }
```

Szerkeszthetőség beállítása

EditingSupport példa - folytatás

```
...
  protected CellEditor getCellEditor(Object element) {
    return editor;
  }

  protected void setValue(Object element, Object value) {
    ((Person) element).email = value.toString();
    getViewer().update(element, null);
  }

  protected Object getValue(Object element) {
    return ((Person) element).email;
  }

}
```

EditingSupport példa - folytatás

```
...
  protected CellEditor getCellEditor(Object element) {
    return editor;
  }

  protected void setValue(Object element, Object value) {
    ((Person) element).email = value.toString();
    getViewer().update(element, null);
  }

  protected Object getValue(Object element) {
    return ((Person) element).email;
  }

}
```

Az egyes elemekhez tartozó CellEditor visszaadása

EditingSupport példa - folytatás

```
...
  protected CellEditor getCellEditor(Object element) {
    return editor;
  }

  protected void setValue(Object element, Object value) {
    ((Person) element).email = value.toString();
    getViewer().update(element, null);
  }

  protected Object get                          t) {
    return ((Person) e
  }

}
```

Az értékek megfelelő beállítása

EditingSupport példa - folytatás

```
...
  protected CellEditor getCellEditor(Object element) {
    return editor;
  }

  protected void setValue(Object element, Object value) {
    ((Person) element).email = value.toString();
    getViewer().update(element, null);
  }

  protected Object getValue(Object element) {
    return ((Person) element).email;
  }

}
```

A beállított érték visszakérése

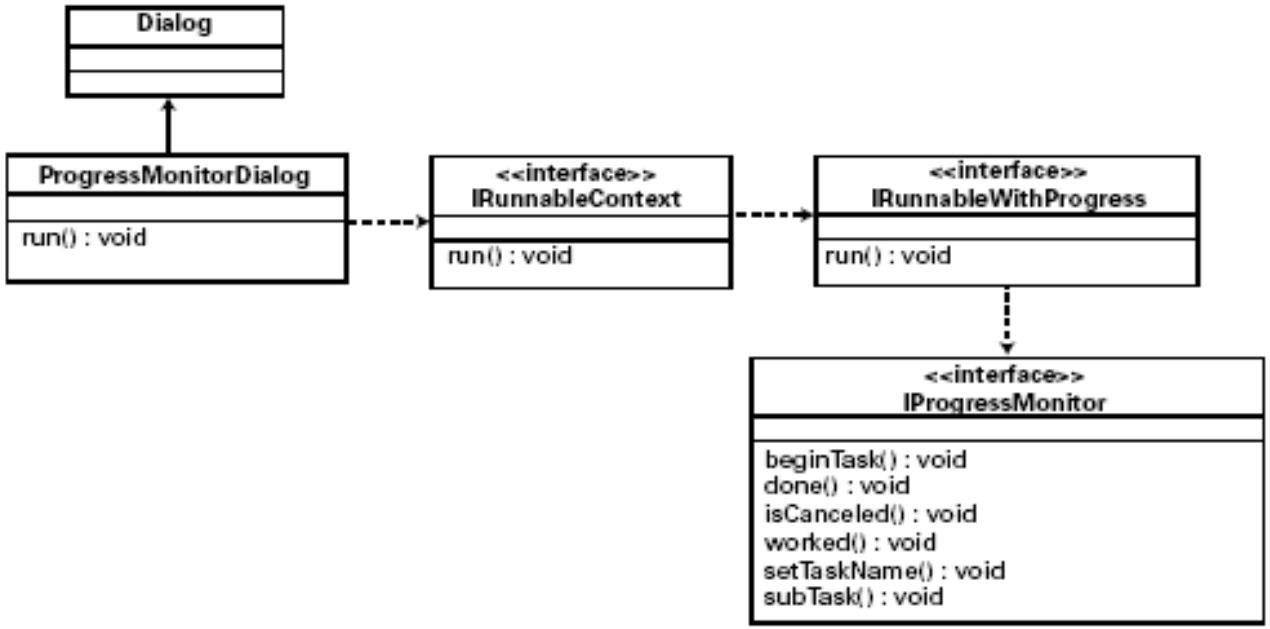
JFace dialógusok és varázslók

JFace dialógusok

- MessageDialog
  - Üzenetek megjelenítése
- ErrorDialog
  - Hibaüzenetek megjelenítése
  - IStatus
    - hiba súlyossága
    - üzenet
    - Exception
- InputDialog
  - Egyszerű szöveg bevitele
  - IInputValidator
    - szöveg ellenőrzése

ProgressDialog

- Háttérben futó munka állapotának megjelenítése



Saját dialógus készítése

- org.eclipse.jface.dialogs.Dialog leszármazott készítése
- Callback metódusok felüldefiniálása
  - `createDialogArea(Composite parent)`
    - Tartalom megjelenítése
  - `createButtonBar(Composite parent)`
    - Gombok elhelyezése
- Címsor megadása
  - configureShell metódus felüldefiniálásával lehetséges

```
protected void configureShell(Shell shell)
  {
    super.configureShell(shell);
    shell.setText("My Dialog Title");
```

Dialógusok beállításai

- Dialógus beállításainak megőrzése (mentés/visszaállítás): DialogSettings osztály
  - `DialogSettings(String)`
  - `put(String, Object)`
  - `save(String)`
  - `load(String)`
  - `get(String)`
  - `get*(String)`

Varázslók

- WizardContainer
  - varázslók gyűjteménye
- Wizard
  - varázsló oldalak
    - gyűjteménye
    - vezérlés (sorrend, mi marad ki, stb.)
  - metódusok
    - `canFinish()`
    - `performCancel()`,`performFinish()`
    - `createPageControls()`

Varázslók

- WizardPage
  - a varázsló oldalai
  - metódusok
    - `getName()`
    - `getNextPage(),getPreviousPage()`
    - `isPageComplete()`
    - `canFlipToNextPage()`

Varázslók

- Feladatok
  - WizardPage
    - Információmegjelenítés
    - Adatbevitel és validáció
  - Wizard
    - Oldalak sorrendjének meghatározása
    - Adatmodell olvasása
    - A végén változtatások végrehajtása
      - Menet közben: bevitt adatok gyűjtése és tárolása
    - Business logic/vezérlés itt jelenhet meg