# Swing – GUI aplikace, události, komponenty, vlákna, postupy

Tomáš Pitner
Masarykova univerzita, Fakulta informatiky

Swing GUI
PV168

## Cíle přednášky, co nás bude zajímat

- Událostmi řízené programování
- Vlákna: principy a nástroje ve Swingu
- Základní komponenty
- Model 1 a MVC ve Swingu

# Co je Swing?

- Prostředí (knihovny, komponenty, POSTUPY) pro psaní javových aplikací běžících na **klientských počítačích** (desktopových aplikací) a majících **grafické uživatelské rozhraní** (GUI).
- Swing je nadstavbou nad **AWT**.
- Existují i nadstavby nad Swing a rámce pro něj – např. SWT, SAF.

## Možnosti oproti AWT

- Kromě základních komponent obsahuje i složitější komponenty - tabulka (**JTable**), strom (**JTree**)
- Umožňuje měnit vzhled a chování aplikací prostřednictvím tzv. **Look and Feel**
- Podporuje pomocné technologie pro nevidomé jako jsou screen readery nebo Braillovy displeje (viz **Java Accessibility**)
- Podporuje snadné programování 2D grafiky (viz **Java2D**)
- Podporuje technologii **Drag and Drop**
- Podporuje snadné vytváření **i18n** a **l10n** aplikací.

# Balíky Swingu

- javax.accessibility
- javax.swing.plaf
- javax.swing.text
- **javax.swing**
- javax.swing.plaf.basic
- javax.swing.text.html
- javax.swing.border
- javax.swing.plaf.metal
- javax.swing.text.html.parser
- javax.swing.colorchooser
- javax.swing.plaf.multi
- javax.swing.text.rtf
- **javax.swing.event**
- javax.swing.plaf.synth
- javax.swing.tree
- javax.swing.**filechooser**
- javax.swing.**table**
- javax.swing.undo

# Info ke Swingu

- Tutoriál na java.sun.com - http://java.sun.com/docs/books/tutorial/uiswing/
- “Swing Book” zdarma - http://www.javafaq.nu/java-allbooks-8.html
- Publikace na O’Reilly Safari - http://safari.oreilly.com (MU má momentálně přístupnou knihu *Java Swing 2nd Edition* - http://proquest.safaribooksonline.com/0596004087)
- Eckel, B. Thinking in Java, http://www.mindview.net/Books/TIJ

# Balíky Swingu

- javax.accessibility
- javax.swing.plaf
- javax.swing.text
- **javax.swing**
- javax.swing.plaf.basic
- javax.swing.text.html
- javax.swing.border
- javax.swing.plaf.metal
- javax.swing.text.html.parser
- javax.swing.colorchooser
- javax.swing.plaf.multi
- javax.swing.text.rtf
- **javax.swing.event**
- javax.swing.plaf.synth
- javax.swing.tree
- javax.swing.**filechooser**
- javax.swing.**table**
- javax.swing.undo

# Start aplikace nad Swing

- Můžeme si zvolit **vzhled** (Look & Feel)
- Potom začneme programově **vytvářet komponenty** GUI
- Inicializace GUI je třeba provádět ve vlákně dispečera událostí, nikoli v hlavním vlákně.
- Zajistíme pomocí **SwingUtilities.invokeLater**

# Volba **Look and Feel**

Možnost nastavit základní “schéma vzhledu”, resp. obsluhy podle:

- javových zvyklostí
- zvyklostí konkrétního OS – tzn. jako nativní aplikace

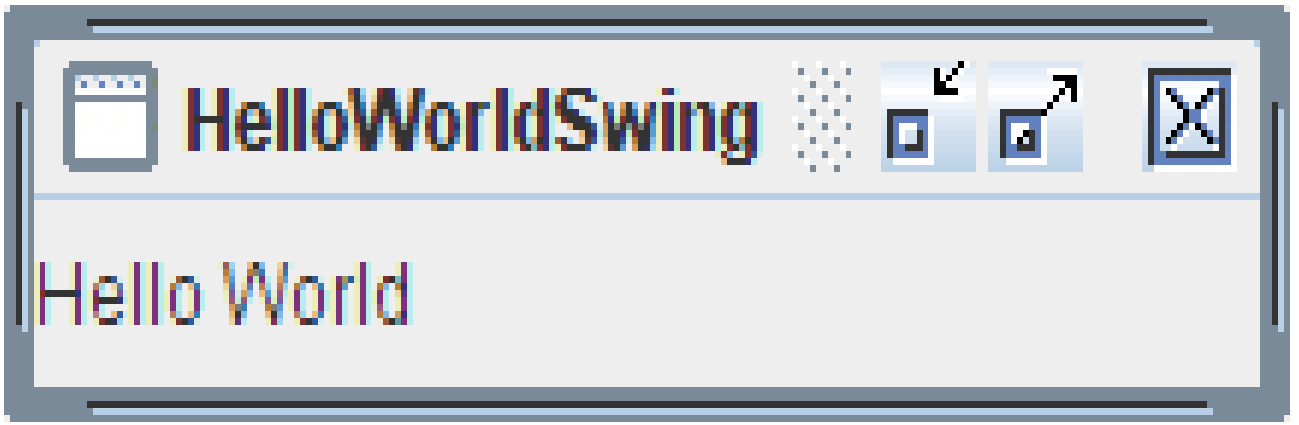
```
public static void main(String[] args) {
    try {
        UIManager.setLookAndFeel(
            UIManager.getCrossPlatformLookAndFeelClassName());
    } catch (Exception e) { }

    // Create and show the GUI...
}
```

# Sestavení GUI

```
import javax.swing.*;
public class HelloWorldSwing {
    private static void createAndShowGUI() {
        JFrame frame = new JFrame("HelloWorldSwing");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel label = new JLabel("Hello World");
        frame.getContentPane().add(label);
        frame.pack();
        frame.setVisible(true);
    }
    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
```

# Sestavení GUI



# Ukončení swingové aplikace

- Pokud má aplikace více vláken, ukončí se až když se ukončí **všechna vlákna** (s výjimkou vláken s nastaveným příznakem daemon), nebo když je zavolána metoda **`System.exit(int)`**.
- 
- Protože vlákno message dispatcheru běží v nekonečné smyčce, aplikace běží dokud není aplikace ukončena voláním této metody.
- Tuto metodu je možné volat buď přímo, nebo je možné pomocí metody **`JFrame.setDefaultCloseOperation(int)`** typicky na hlavním okně zajistit její automatické zavolání při zavření tohoto okna.

*Petr Adámek, kore/wiki/index.php/**Swing***

# Událostmi řízené programování

- Základní koncept řízení toku programu nad Swing
- Událostmi se rozumí **akce uživatele** prostřednictvím periférií (myš, klávesnice,…) **nad komponentami GUI** – *nízkoúrovňové události*
- *Vysokoúrovňové události* jsou “druhotné” vytvořené komponentami

- Události (tedy jejich popisy) se řadí do fronty **message dispatcheru**
- Ošetření (reakce na událost) se neprovádí v hlavním vlákně aplikace
- Nemělo by trvat dlouho
- Dlouhé výpočty je třeba delegovat do jiných vláken (**SwingWorker**)

*Petr Adámek, kore/wiki/index.php/**Swing***

# Ošetření události

- Vytvořit tzv. **posluchače** (EventListener), který bude na danou událost reagovat.
- Posluchač se pak musí **zaregistrovat** jako příjemce události u příslušné komponenty.

# Ošetření události

```
// Vytvoříme instanci tlačítka
JButton button = new Jbutton("Moje");

// Vytvoříme posluchače události
ActionListener actionListener = new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Zobrazíme box s informací o stisknutí tlačítka
        JOptionPane.showMessageDialog(null,"Stisknuto tlačítko:"
          + e.getActionCommand());
        }
    };

// Zaregistrujeme posluchače u kompon., kde se událost generuje
button.addActionListener(actionListener);
```

# Kód měnící GUI

- Musí být spuštěn ve **vláknu message dispatcheru**, protože metody ve Swing nejsou jinak synchronizovány.
- Týká se i prvotní konstrukce GUI (hlavní okno, menu,…) i změn – např. zneviditelnění tlačítka, položky menu
- Lze provést pomocí SwingUtilities

# Spouštění ve vlákně message dispatcheru

Metoda **`invokeLater`** - spustí ve vlákně message dispatcheru **asynchronně** (ne hned a nečeká na ukončení **`run()`**)

Signatura: `public static void invokeLater(Runnable doRun)`

```
Runnable doHelloWorld = new Runnable() {
    public void run() {
            System.out.println("Hello World on "+
            Thread.currentThread());
    }
};
SwingUtilities.invokeLater(doHelloWorld);
System.out.println("This might well be
    displayed before the other message.");
```

# Spouštění ve vlákně message dispatcheru

Metoda **`invokeAndWait`** - spustí ve vlákně message dispatcheru **synchronně:**

- Ne hned, ale až po ukončení čekajících vláken
- Pak počká na ukončení **`run()`**

Signatura:

```
public static void invokeAndWait(Runnable doRun)
                    throws InterruptedException,
                           InvocationTargetException
```

# Spouštění ve vlákně message dispatcheru

```
final Runnable doHelloWorld = new Runnable() {
    public void run() {
        System.out.println("Hello World on " +
            Thread.currentThread());
    }
};
Thread appThread = new Thread() {
    public void run() {
        try { SwingUtilities.invokeAndWait(doHelloWorld);
        } catch (Exception e) { e.printStackTrace(); }
        System.out.println("Finished on "+Thread.currentThread());
      }
 };
 appThread.start();
```

# Dlouhé výpočty

- Ukládání a čtení souborů, ze sítě, dlouhé kalkulace...
- Spouštíme naproti tomu v separátním vlákně (vláknech), nejlépe pomocí *SwingWorker*

```
worker = new MySwingWorker();
worker.addPropertyChangeListener(progressListener);
worker.execute();
```

*příklad viz Petr Adámek, kore/wiki*

# Model 1

- Zjednodušená verze **Model-View-Controller (MVC)**
- Chybí kontroler, jen **oddělený obsah** (model, data) od **prezentace**

```
worker = new MySwingWorker();
worker.addPropertyChangeListener(progressListener);
worker.execute();
```

*příklad viz Petr Adámek, kore/wiki*

# Internacionalizace

- Podpora v NetBeans
- V příslušném (hlavním) **balíku** aplikace
- Vytvořit ResourceBundle pomocí: New/Other/**Java Properties**
- Přidat všechna očekávaná národní prostředí: *rightclick*/**Add Locale**
- Nad .properties souborem: *rightclick*/**Open** (ne Edit!)

- Nad inspektorem objektů – u **vlastnosti text** kliknout na … a zvolit "from ResourceBundle" a vybrat
- Podobně lze vybrat i **horkou klávesu** pro aktivaci, tooltip atd.

# JTable a TableModel

- Implementuje **Model 1**, odděluje ***data*** od ***prezentace*** (komponenty)

- *Tabulka* je typu javax.swing.**JTable**;
- *Model* implementuje rozhraní **TableModel** nebo rozšiřuje javax.swing.table.**AbstractTableModel**
-
- Podobně funguje např. **JList** a **JTree**...

# Příklad TableModelu

```
class MyTableModel extends AbstractTableModel {

    private String[] columnNames = {"First Name",
       "Last Name", "Sport", "# of Years", "Vegetarian"};
    private Object[][] data = {
            {"Mary", "Campione","Snowboarding", new
             Integer(5), new Boolean(false)},
            {"Alison", "Huml",
             "Rowing", new Integer(3), new Boolean(true)}
    };

    public int getColumnCount() {
```

# Příklad TableModelu

```
public int getRowCount() {
    return data.length;
}

public String getColumnName(int col) {
    return columnNames[col];
}

public Object getValueAt(int row, int col) {
    return data[row][col];
}
```

# Příklad TableModelu

```
public Class getColumnClass(int c) {
    return getValueAt(0, c).getClass();
}

public boolean isCellEditable(int row, int col) {
    //Note that the data/cell address is constant,
    //no matter where the cell appears onscreen.
    //...
}
```

# Příklad TableModelu

```
public void setValueAt(Object value, int row, int col) {
    // ...
    data[row][col] = value;
    fireTableCellUpdated(row, col);
}
```