Grafické uživatelské rozhraní Swing

Obsah

Možnosti

Grafická uživatelská rozhraní v Javě

Swing rozšiřuje původní knihovnu pro uživatelské rozhraní, která se nazývala AWT.

AWT — mezivrstva mezi programem a nativními komponentami uživatelského rozhraní daného operačního systému

Swing — vlastní knihovna komponent, komponenty OS nevyužívá

AWT (Abstract Windowing Toolkit)

výhody — rychlejší, aplikace vypadají jako nativní

nevýhody — starší, hůře navržen, menší repertoár komponent

Swing

výhody — kvalitní návrh, i pokročilé komponenty, vzhled a chování možno nastavovat (Look and Feel), přístupnost (Accessibility), Drag&Drop, Java 2D, podpora i18n a l10n

nevýhody — pomalejší a hůře přizpůsobený nativnímu prostředí (pomalu přestává platit...)

Principy

Řízení událostmi

Event-driven programming

- Jednotlivé komponenty programu reagují na události:

  nízkoúrovňové — klikání a pohyby myší, stisky tlačítek

  vysokoúrovňové — "logické", generované jinou komponentou - např. "stisk" tlačítka v dialogu

- Obsluha události je asynchronní vůči jiným vláknům - např. program může souběžně ukládat na disk a editovat text.
- Události se řadí do *fronty událostí* a jsou obsluhovány vláknem *message dispatcher*.
- Na delší akce je proto třeba vytvořit vlákno separátní, aby se neblokovalo ošetřování dalších událostí.

Ošetření události

zdrojová komponenta	kde událost vznikne - na ni je třeba "zavěsit" posluchače události -> JButton
posluchač události	objekt, který pomocí "call-back" metody reaguje na vzniklou událost - tato metoda je vyvolána -> ActionListener
kód posluchače	většinou implementujeme rozhraní ActionListener přímo na místě jako anonymní vnitřní třídu, jejíž instance má pak přístup k obsahu "mateřské" instance třídy, v niž je vnořena

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

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

// Zaregistrujeme posluchače u komponenty, která událost může generovat
button.addActionListener(actionListener);
```

Vlákna

Jak tušíte, ve světě událostmi řízených programů to nebude jednoduché...

Je třeba dodržovat následující:

- Všechny operace s komponentami GUI *musí* být prováděny ve vlákně message dispatcher.
- Časově náročné operace (např. načítání dat, výpočty, apod.) provádějme v *jiném* vlákně než je message dispatcher.

Ve vlákně message dispatcher

Pomocí metody `SwingUtilities.invokeAndWait` nebo `SwingUtilities.invokeLater` - podobně, jako když vytváříme vlákna. Kterou kdy?

> I want to update a Swing component *but I'm not in a callback* (!!!) If I want the update to happen immediately (perhaps for a progress bar component) then I'd use `invokeAndWait`. If I don't care when the update occurs, I'd use `invokeLater`. (from Interview Helper [http://www.interviewhelper.org/Art/2219/10/Why-would-you-use-SwingUtilities-invokeAndWait-or-SwingUtilities-invokeLater.html])