TimeJuggler - Java kalendář

– semestrální práce jako součást předmětu X36PMT (projekt v týmu)

Vypracovali:
Bc. Ladislav Vitásek (vitasl1@fel.cvut.cz)
Bc. Jan Struž (struzj1@fel.cvut.cz)
Bc. Jiří Holý (holyj3@fel.cvut.cz)
FEL ČVUT 2007

Obsah

1 O tomto dokumentu

Tento dokument slouží jako doplněk k programátorské dokumentaci, aby bylo možné snadněji navázat při dalším vývoji aplikace.

Aktuální znění tohoto dokumentu, včetně poslední verze zdrojových kódů lze nalézt v Subversion repository na [1].

2 Pokyny a zadání

Navrhněte, implementujte a otestuje přenositelnou aplikaci kalendář tak, aby odstranila nevýhody existujících kalendářových aplikaci, mezi které patří například:
- špatná přenositelnost aplikací,
- nemožnost archivovat a zálohovat data kalendáře, a do archivů zpětně a snadno nahlížet,
- nahlížet do kalendářů nouzově v textovém režimu (aplikace používají vestavěnou databázi a grafické uživatelské rozhraní),
- špatný vzhled a uživatelskou pohodlnost při používání, absenci barev a moderních prvků uživatelských rozhraní,
- snadnou synchronizaci s mobilním zařízením,
- provázání seznamu úkolů s kalendářem a přehledné zobrazování tzv. "deadlinů",
- snadnou manipulaci s událostmi pomocí myši a klávesnice.

3 Architektura a použité komponenty implementované aplikace

Aplikace byla navržena a implementována v jazyce Java pod JDK verze 6 za pomoci architektury Swing. K ukládání dat je použita databáze Derby [10], pro manipulaci s ICS soubory (standardní formát pro výměnu kalendářů) je použita knihovna iCal4j [9].

4 Swing Application Framework (application.*)

Swing Application Framework (SAF) tvoří základní kámen aplikace. Zprostředkovává základní architekturu a sadu služeb nutných k provozu Swing aplikace, které prakticky každá aplikace musí implementovat. Jedná se o základní cyklus (start, inicializace, startup, ready, exit, shutdown), událostní zpracování UI komponent, lokalizování (překlady aplikace do jiných jazyků, adaptace na různý operační systém), správa vláken (déle trvající úlohy běžící na pozadí a jejich monitoring), jednoduché ukládání stavových informací (uživatelské nastavení).

Tento stručný popis si nekladl za cíl kompletně detailně popsat jednotlivé vlastnosti frameworku, ale má dát čtenáři základní přehled o možnostech a nadhled do zdrojových kódů aplikace. Více informací, včetně jednoduchých ukázkových příkladů, lze nalézt na webové stránce projektu [2], prezentaci z JavaOne [3], případně v tutoriálu [4].

Poznámka: V současnosti se jedná o prototyp, který slouží jako referenční implementace Java Specification Request (JSR 296) a je stále průběžně vyvíjen a měněn. Neočekávají se ovšem již nikterak zásadní změny (většinou jde o opravu bugů). Jedinou význačnou chystanou zpětně nekompatibilní úpravou je převedení hlavní package tříd `Application.*` do `javax.swing.application` . Tento krok je plánován spolu s integrací tohoto frameworku do JDK7.

5 Komponenty SwingX (org.jdesktop.*)

K základním komponentám Java Swing byly použity navíc další z open source knihovny SwingX [5], které přidávají více jak uživatelského, tak i programátorského komfortu.

Jedná se o

- tabulky (`JXTable`),
- mini kalendářová komponenta (`JXMonthView`),
- statusbar (`JXStatusbar`),
- layout manager pro rozmístění hlavních komponent (`JXMultiSplitPane`)
- klasický seznam položek umožňující filtrování a zvýrazňování (`JXList`)

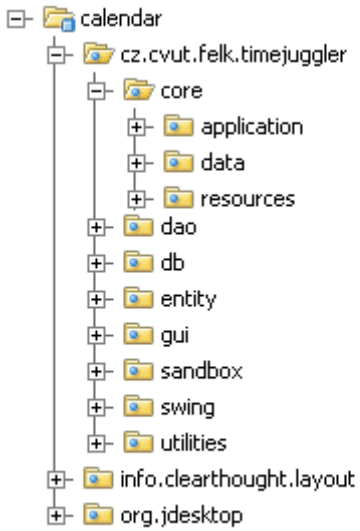
Z knihovny SwingX bylo taktéž použito rozšíření základní třídy `JFrame` (`JXFrame`), které navíc obsahuje metody pro blokování uživatelského vstupu při déle trvající činnosti (když na pozadí běží např. dotahování dat z databáze apod.).

6 Komponenty JGoodies (com.jgoodies.*)

JGoodies Swing Suite komponenty [6] byly použity ke zjednodušení vývoje a zvýšení čitelnosti kódu. Binding framework je využíván k lepšímu internímu propojení – mapování – dat s komponentami (více lze nalézt zde třeba na [7]). `FormLayout` pak k efektnímu a korektnímu rozmístění formulářových komponent v dialozích (bezproblémová změna velikosti dialogu).

7 Základní popis rozdělení package aplikace

Aktuální strukturu zdrojových kódů aplikace je možné vidět na následujícím obrázku.



Obrázek 1 - Struktura zdrojových kódů

Package `cz.cvut.felk.timejuggler`

- `core` – obsahuje důležité součásti operující nad SAF a životním cyklem aplikace
  - `application` – správa neodchycených vyjímek
  - `data` – správa globálních dat – správa perzistenční vrstvy
  - `resources` – soubory s lokalizací a ikony hlavních akcí
- `dao` – pomocný package s třídami pro ladění kalendářové komponenty
- `db` – správa datového úložiště
- `entity` – entitní třídy pro kalendářovou komponentu
- `gui` – grafické uživatelské rozhraní – dialogy aplikace a správa hlavních komponent (menu, toolbar, statusbar, seznam událostí, seznam tasků)
- `sandbox` – "pískoviště" – odkladiště tříd pro možné budoucí použití, pokusné testování
- `swing` – UI komponenty rozšiřující základní Java Swing komponenty (např. `ColorComboBox` aj.)
- utilities – třídy s pomocnými metodami (přehrávání zvuků, otevírání webového browseru apod.)

Package `info.clearthought.layout` obsahuje layout manager `TableLayout` [8].

8 Logování

Aplikace využívá standardní prostředky pro logování z package `java.util.logging`, které rozeznává tyto základní úrovně:

- SEVERE (nejvyšší úroveň)
- WARNING
- INFO
- CONFIG
- FINE
- FINER
- FINEST (nejnižší úroveň)

Veškeré výstupy z logovacího mechanismu jsou směřovány jak do konzole, tak do souboru pojmenovaném `timejuggler.log`. Výchozím nastavením je úroveň WARNING. Předdefinovanou nižší úroveň INFO (a tedy i podrobnější výpisy) lze zapnout při startu aplikace pomocí přepínače debug (viz Parametry pro spuštění aplikace).

Výchozí i 'debug' nastavení logování a lze upravit v souborech `logdefault.properties` a `logdebug.properties`, které jsou umístěny v kořenovém adresáři zdrojových kódů `src` (po kompilaci jsou umístěny v hlavním JAR souboru).

9 Moduly aplikace

- databázový modul
- uživatelské rozhraní
- kalendářová komponenta

10 Databázový modul (cz.cvut.felk.timejuggler.db.*)

Tento modul slouží k přímé komunikaci s databází. V našem případě s databází DerbyDB. Umožňuje ukládání a čtení dat z databáze, exportování a importování ze souborů *.ics*. Struktuře ICS se věnuje dokument RFC2445, podle kterého byl také vytvořen datový model. S výhodou byla využita již existující knihovna *iCal4j* [9], která je implementací zmíněného RFC v Javě. Tato knihovna zajišťuje importování souborů ICS do vlastní vnitřní formy. Pomocí svých objektů umožňuje vytvářet a modifikovat kalendářová data, události a také je exportovat do souborů ICS. Úkolem databázového modulu je mj. mapování objektů *iCal4j* na databázové objekty, s kterými pracuje aplikace.

10.1 Kalendářové objekty

Celá struktura kalendáře je složena z kalendářových objektů (komponent – viz níže), které mají svoje vlastnosti (Property). Kalendář může obsahovat více takových objektů. Některé vlastnosti jsou nepovinné, jiné se mohou objevit vícekrát v téže komponentě, a mohou být v databázi i složitější strukturou (např. *RepetitionRule* - pravidla pro opakování událostí). Vlastnosti mohou mít navíc určitou množinu parametrů, ty však zatím databázovém modelu nejsou implementovány.

10.1.1 Třída VCalendar

Reprezentuje v databázi pojmenovaný kalendář.

10.1.2 VEvent a VToDo – implementovány třídou EventTask

Třída *EventTask*, umožňuje přechod od typu *VEvent* k *VToDo* (a naopak) a jednotnou manipulaci. V databázi se jedná o rozdílné tabulky.

10.1.3 CalComponent

Předek tříd *VEvent* a *VToDo* (*EventTask*) - obsahuje všechny společné vlastnosti.

10.1.4 VFreeBusy

Označení časového úseku jako „volný“ či „obsazený“. Neimplementováno.

10.1.5 VAlarm

Možnost informování uživatele určitý časový interval před důležitou událostí. Alarm (více) může být přiřazen jak událostím, tak úkolům. Podle RFC také vlastnosti *Attendee* (na to db.model není připraven). Neimplementováno.

10.1.6 VJournal

Pro shrnutí informací popisující jednotlivý den. Neimplementováno.

10.1.7 VTimeZone

Pro popis časové zóny, kde a kdy se události konají. Neimplementováno.

10.2 Entity

Z důvodu přísného oddělení přístupu aplikace do databáze a manipulace s objekty, které data reprezentují požívá aplikace vícevrstvový model. Datový model má jednu sadu tříd, které obsahují implemetaci přístupu do databáze a navíc *interfacy*, se kterými se pracuje v uživatelském rozhraní.

Všechna tato rozhraní jsou pojmenována jako <název-db-třídy> a končí na -*Entity.*

10.3 Databázové mapování

10.3.1 Třída DbElement

Základní třída všech objektů, které lze ukládat do databáze (mají databázové ID). Po vytvoření objektu zděděného z třídy *DbElement* je třeba zavolat metodu *saveOrUpdate*, která se postará o vložení databáze (včetně vnořených objektů). V případě změny instance stačí tuto metodu zavolat znovu. Tentokrát se provede automaticky pouze update. Pro vymazání z databáze slouží metoda *delete*.

10.3.2 Třída JDBCTemplate

Třída JDBCTemplate využívá návrhového vzoru TemplateMethod, stará se o všechny náležitosti ukládání a čtení dat z databáze. Je abstraktní, aplikace využívá jejího potomka *TimeJugglerJDBCTemplate*. Využívá se také možnosti transakcí, tedy všechny operace provedené nad instancí této třídy jsou označeny jako jedna transakce. Např. nastavíme všechny vnitřní hodnoty objektu *EventTask*, včetně vnitřních tříd, a metodu *saveOrUpdate* nad ním zavoláme pouze jednou.

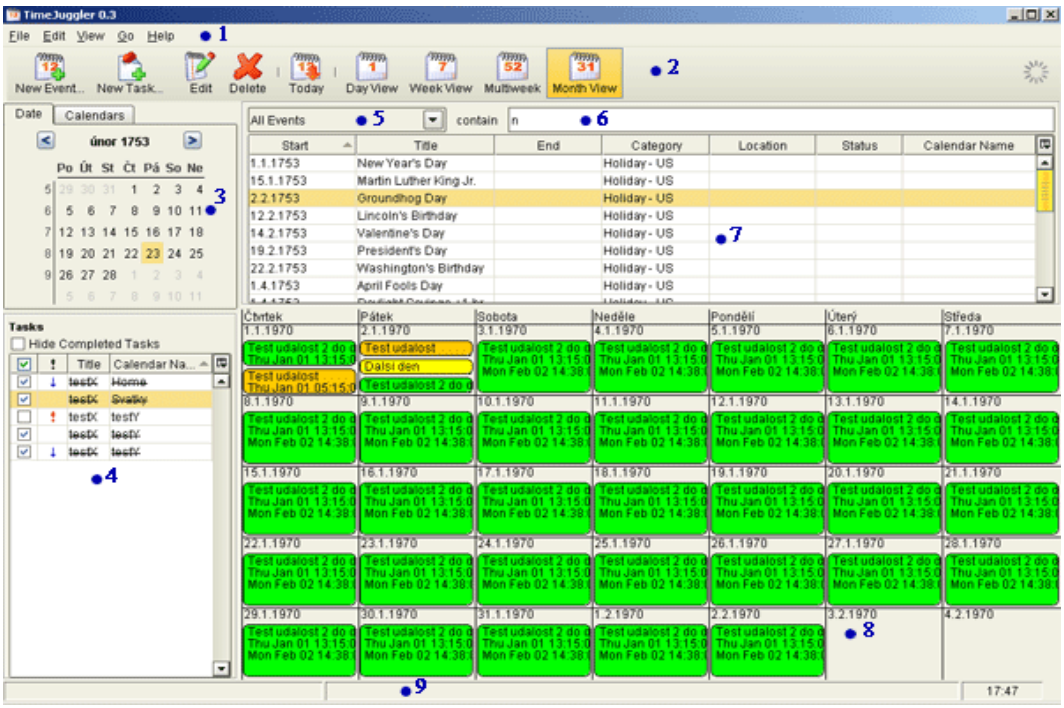
10.4 SQL

Všechny třídy jako samostatné objekty mají plně implementované *insert*, *update* a *delete*. Avšak některé dotazy typu *select* je třeba doplnit. Z důvodu rychlosti při pracování s aplikací by při načítání objektů mělo být do databáze přistupováno co nejméně často. Tedy kupř. načtení najednou všech údajů, které s sebou nese objekt *EventTask* včetně možných vnořených tabulek, není vůbec jednoduché.

11 Modul uživatelského rozhraní

11.1 Základní GUI aplikace

GUI pro uživatele se snaží být co nejjednodušší s maximálním uživatelským pohodlím.

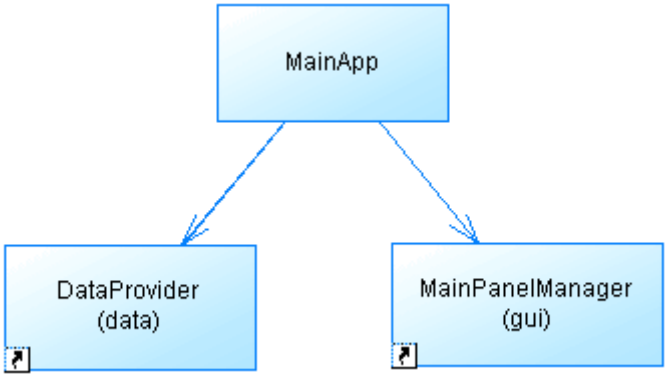


Legenda:

1. Hlavní menu
2. Toolbar
3. Monthview – zobrazení aktuálně zvoleného měsíce s vyznačenými důležitými dny (dle událostí)
4. Seznam úkolů
5. Filtr událostí
6. Fulltextové vyhledávání v seznamu událostí
7. Seznam událostí
8. Calendar Grid – kalendářová komponenta
9. Statusbar

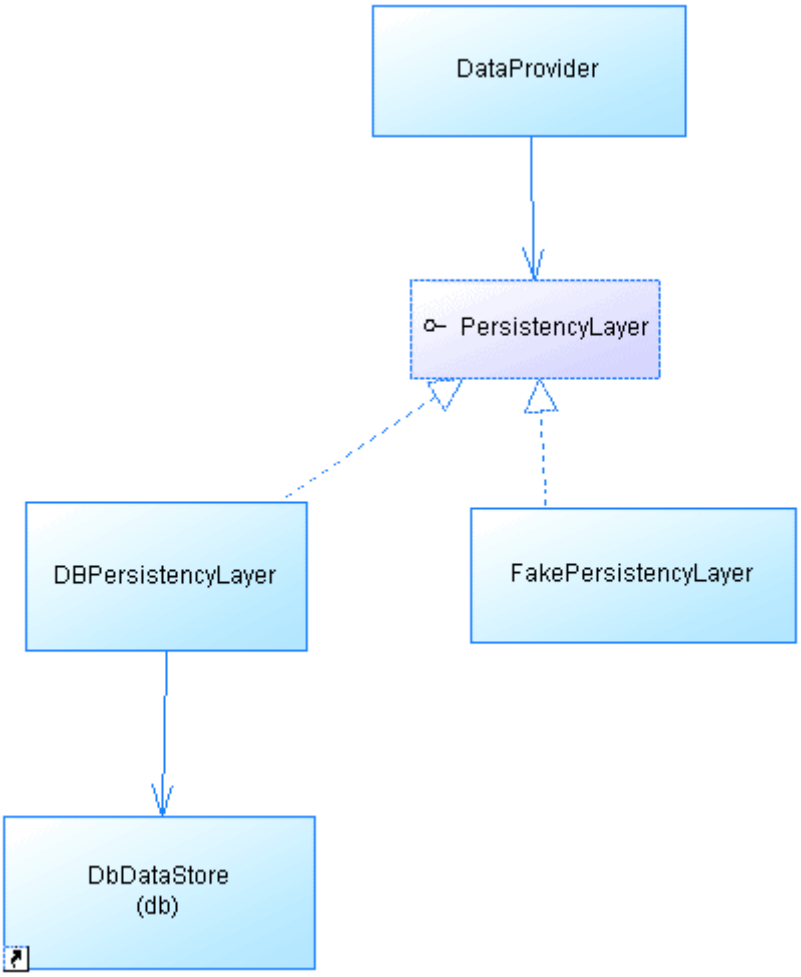
11.2 Aplikační vrstvy

Hlavní spouštěcí třída `MainApp`, která implementuje rozhraní `Application` SAF (která je singletonem a tedy je dostupná odkudkoliv z aplikace) udržuje reference na instanci třídy `MainPanelManager` a `DataProvider` (viz zjednodušené UML Class diagramy na obrázku). Pomocí `MainPanelManageru` lze dále přistupovat ke GUI komponentám aplikace (, `MenuManager`, `StatusBarManager`, `EventsListManager` atd.)



Obrázek 2 – Reference na DataProvider a MainPanelManager

`DataProvider` je zodpovědný za přístup k datům v datovém úložišti. Nezávisí na tom, kde datové úložiště fyzicky leží – perzistenční vrstvu lze zaměňovat. Jedinou povinností je implementovat rozhraní `PersistencyLayer` (viz obr).



Obrázek 3 - DataProvider - získávání dat

Takto lze například simulovat, přístup k databázi za pomoci třídy `FakePersistencyLayer`, která byla vytvořena čistě jen pro testovací účely.

11.3 Spouštění

Ke spuštění aplikace je nutné mít nainstalovánu Javu verze 6. Na operačním systému MS Windows stačí spustit `timejuggler.exe`, na Linuxu je možné kalendář také využívat.

Aplikaci lze vždy spustit takto:

java –jar timejuggler.jar

11.4 Parametry pro spuštění aplikace

Použití: `timejuggler [-přepínače] [soubor(y)]`

možnosti:

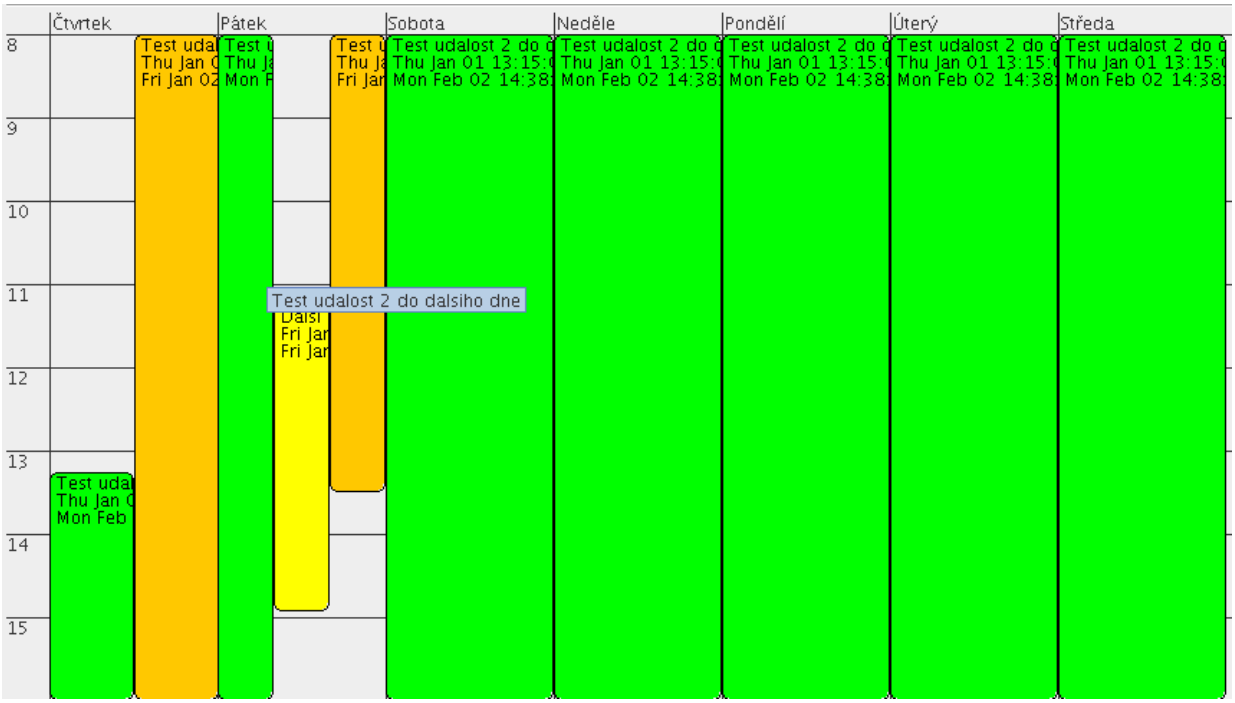
- `-h, --help` – zobrazení
- `-v, --version` – zobrazení informace o verzi aplikace
- `-d, --debug` – zapnutí interní úroveň logování na úroveň INFO
- soubor(y) – cesta k souborům k otevření (prozatím neimplementováno)

12 Modul kalendářové komponenty

Zpočátku jsme uvažovali o využití již hotové komponenty. Naneštěstí nalezená implementace (migcalendar) není volně dostupná zdarma. Proto jsme se pokusili o vytvoření vlastní komponenty.

12.1 CalendarGrid

Tato vizuální komponenta si klade za cíl zobrazovat události i úkoly v mřížce a zároveň nabízet informace s tím spojené.



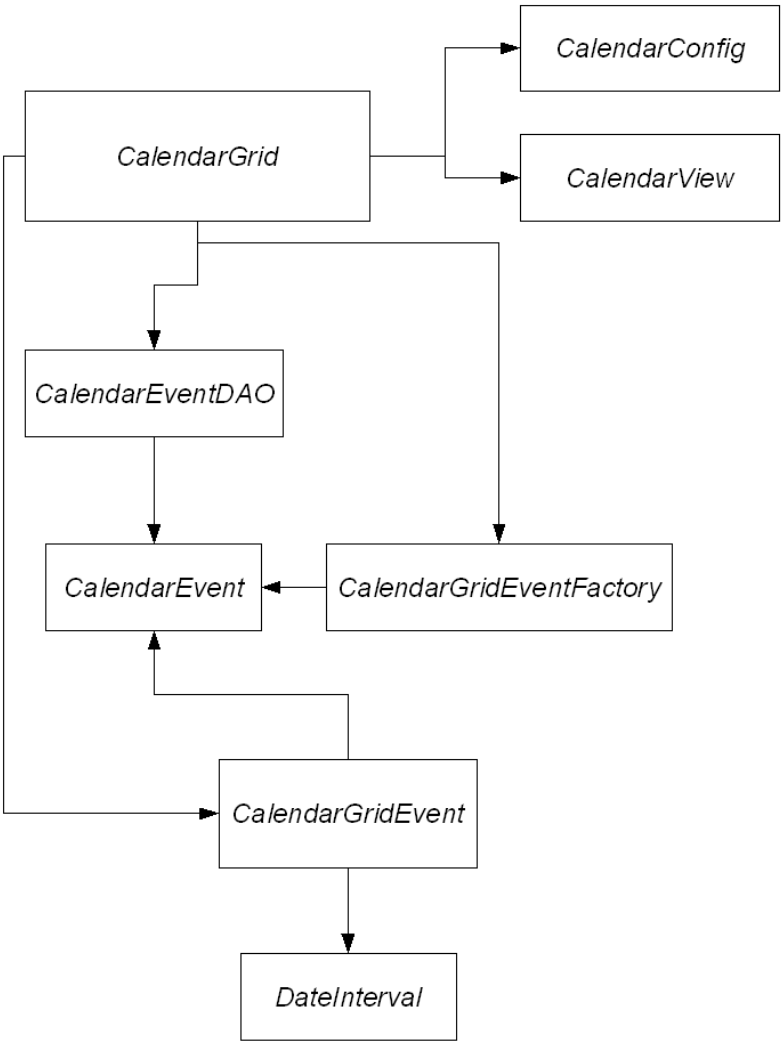
O

brázek 4 - CalendarGrid

12.2 Implementace CalendarGrid

Komponenta byla navržena jako samostatný objekt, snahou bylo veškerý možný kontakt s prostředím oddělit pomocí interface. Grafické komponenty jsou v hierarchii tříd potomky JComponent, aby uměly základní operace obvyklé prostředí.

Hlavní třídou a zároveň onou vizuální komponentou je třída *CalendarGrid*. Její vlastní nastavení probíhá pomocí metod *setCalendarConfig* (nastavení konfigurace kalendáře) a *setCalendarView* (jak bude kalendář vypadat). Třída získává data pro zobrazení přes *CalendarEventDAO*, ty pak dělí podle potřeby na různé časové intervaly (z důvodu tvorby více různých vizuálních komponent, reprezentujících právě jednu jedinou událost). O vlastní tvorbu vizuálních komponent představujících událostí se stará třída *CalendarGridEventFactory*. Třída *CalendarGrid* se poté postará o správné rozmístění komponent představujících události a vykreslení vlastní kalendářové mřížky (pozadí).



Obrázek 5 - Hrubé schéma tříd a rozhraní

Obnovení událostí se děje pomocí metody *refreshCalendarEvents*. Jedinou informaci, kterou

mřížka kalendáře dokáže poskytovat, je datum podle souřadnic, k čemuž slouží metoda *getDateByPosition*.

12.3 CalendarGridEvent

Vizuální komponenta představující právě jednu událost (*CalendarEvent*). Stará se o vlastní vykreslení události. O její životní cyklus se v běžném případě stará *CalendarGrid* (resp. *CalendarGridEventFactory*). Více takovýchto komponent může stále představovat jednu událost, ale rozdělenou do různých časových intervalů. To proto, že událost se může například opakovat a pak je každá z těchto komponent právě jedním z opakování. Ve vizuálním prostředí se tak jednodušeji zobrazí dvě jednoduché komponenty, než jedna složitá.

12.4 CalendarGridEventFactory

Tovární rozhraní vyrábějící jednotlivé vizuální *CalendarGridEvent*. Podle implementace pak může vytvářet různé potomky *CalendarGridEvent* a tím jednoduše rozlišit jejich možnosti, nebo jen způsob vykreslování.

12.5 CalendarView

Enumerace typu zobrazení den, týden, multi-týden, měsíc.

12.6 CalendarConfig

Umožňuje různá nastavení kalendáře. V tuto chvíli hodiny, v kolik den začíná a končí.

12.7 CalendarEvent

Datová beana, důležitý je zde časový interval *startDate – endDate.*

12.8 CalendarEventDAO

Datový interface vyhledávající *CalendarEvent* (*s*) v zadaném datovém intervalu.

13 Další možné použití

Protože se jedná pouze o vizuální komponentu, je možná celá řada dalších rozšíření. Jedním z nejžádanějších je asi operace Drag&Drop. Protože všechny grafické třídy jsou potomky *JComponent*, lze tak vše normálně podřídit běžné tvorbě. Při operaci Drag poskytují komponenty zobrazující události (*CalendarGridEvent*) přístup k zobrazované události i jejím umístění v kalendářové mřížce. Naopak operace Drop může poskytnout souřadnice v mřížce, které pak lze převést na datum. Ovšem žádná z operací by mřížku neměla měnit, protože ta data pouze zobrazuje, nikoliv mění či jinak spravuje.

14 Nedokončené části

Přes veškerou snahu se nám z důvodu časové tísně nepodařilo aplikaci v plné šíři dokončit. Přesto lze říci, že podstatná část komponent, na kterou lze dále navázat, je dokončena.

14.1 Aplikační část

Propojení zbývajících dialogů a nabídek s entitami datového modelu (vytváření/editace událostí/úkolů). Dialogy jsou dokončeny kromě podpory validace. Podpora pro filtrování seznamu událostí (combobox vedle řádku pro fulltextové vyhledávání).

14.2 Datová část

Sjednotit datové typy (SQL) vlastností, které lze pro zjednodušení ukládat jako konstanty (podle Mozilla Sunbird). Přidat možnost ukládání parametrů do datového modelu (důležité parametry).

Implementovat management časových pásem a alarmů.

Umožnit rozšířené filtrování událostí podle zadaných kritérií.

Dokončit import/export událostí (zejména opakovací pravidla, timezones, alarmy). Při importu je třeba nejprve vložit do databáze případné nové kategorie nalezené v souboru.

Poznámka: vlastnost DTSTART by měla náležet do tabulky Period (jako první záznam pro danou událost, neměla by se ukládat explicitně zvlášť, ale spíše pouze jako „cachovaná“ hodnota).

14.3 Kalendářová komponenta

Propojení zobrazovaných událostí s objekty používanými v aplikaci.

Drag&Drop událostí a úkolů.

15 Odkazy a zdroje

[1] Subversion repository – http://svn.wordrider.net/svn/timejuggler/trunk/
jméno a heslo:wordrider
[2] The Swing Application Framework – https://appframework.dev.java.net
[3] Using the Swing Application Framework –
http://java.sun.com/developer/technicalArticles/javase/swingappfr
[4] Swing Application Framework – JavaOne 2007 presentation
http://weblogs.java.net/blog/hansmuller/archive/ts-3492-final.pdf
[5] SwingX components – http://swinglabs.org/
[6] JGoodies komponenty – http://www.jgoodies.com
[7] JGoodies tutorial – http://www.javalobby.org/java/forums/t17672
[8] Komponenta TableLayout – https://tablelayout.dev.java.net/
[9] Knihovna iCal4j - http://ical4j.sourceforge.net/
[10] Databáze DerbyDB - http://db.apache.org/derby/