Pavel Klavík

CompEvid

Programátorská reference

http://pavel.klavik.cz/compevid

Úvod

CompEvid je program na vedení evidence nedokončené výroby. V programu si vedete evidenci firem a přidáváte do ní doklady. Každý doklad může být rozúčtován na několik zakázek. V programu potom můžete procházet jak seznam dokladů, tak seznam zakázek. Každá zakázka se totiž může skládat z více dokladů. Tento program byl vytvořen na míru, aby přesně odpovídat potřebám. Proto některé veci jsou záměrně zjednodušené nebo nastavené pro danou situaci.

Tento text by měl sloužit jako programátorská dokumentace. Snažil jsem se v něm popsat základní strukturu a architekturu projektu. Program byl napsán v C++ s použitím knihovny Qt 4. Předpokládá se velice slušná znalost obojího. Pokud si chcete dostudovat Qt, určitě se vám velice bude hodit výborná dokumentace, kterou naleznete na domovských stránkách Trollů: `http://doc.trolltech.com/`.

CompEvid byl vytvořen jako zápočtový program na předmět programování a je uvolněn pod licencí GNU GPL. Byl testován pod operačními systémy Linux a Windows. Součástí dokumentace je také uživatelská reference. Doporučuji se nejprve s programem zevrubně seznámit. Lze ji stáhnout na `http://pavel.klavik.cz/compevid`.

TODO: Pomocí tohoto symbolu označuji podstatné části o tom, co by se mělo předělat a upravit, či co mi nepřijde uplně dostatečné. Pokud se mi podaří dané problémy vyřešit, budou tyto části postupně mizet. Ideálně po čase už ani žádnou nenaleznete.

Historie

Je tomu více než dva roky zpátky, co na Vánoce 2006 vznikla první verze původního programu CompEvid. Byl to výsledek zhruba dvou týdnů práce. Program byl vytvořen pomocí dosti „obskurních“ technologií C++ a MFC. Zhruba půl roku zpátky bylo potřeba provést některé jednoduché úpravy ve zdrojových kódech. Avšak vzhledem ke kvalitě napsaných zdrojových kódů a chybějící dokumentaci jsem se rozhodl program vystavět na zelené louce. Není pravda, že bych nedokázal do programu přidat pár nových kolonek a upravit tisk. Ale program nevyhovoval po celé řadě dalších stránek. Jeho ovládání bylo pochybné a nepracovalo se s ním příjemně. Aplikace byla špatně navržena a proto nereflektovala přesně potřeby klienta.

CompEvid byla moje první aplikace s grafickým rozhraním pod Linuxem. Potřeboval jsem však, aby grafické rozhraní běželo na Windowsech. Pro knihovnu Qt jsem se rozhodl právě z těchto důvodů. Navíc s ní v té době už další dobu pracoval kamarád a mohl jsem s ním spoustu věcí prokonzultovat. Knihovnu Qt považuji za výbornou po všech stránkách, krásně se používá a programy chodí rychle a bezproblémově na různých operačních systémech. Navíc má naprosto dokonale vytvořenou dokumentaci, ve které člověk během chvilky najde přesně to, co potřebuje.

Základní informace

Začněme zprvu ve stručnosti, o základních třídách a konceptech programu CompEvid.

TODO: Celý kód by potřeboval hodně začistit, chtělo by to ujednotit názvy funkcí a proměnných, sdružit podobné věci k sobě. Tedy hromada nudné práce, která by ale opravdu byla potřeba.

Použité třídy

Hlavní struktura programu je tvořena dvěma třídami: `MainWindow` a `DataModel`. Kolem nich se ohýbá celá struktura aplikace. Již název vypovídá o jejich funkci. `MainWindow` zastřešuje hlavní okno, jeho ovládací prvky a některé funkce. Naproti tomu `DataModel` se stará o data firmy, zastřešuje jejich úpravy, získávání a částečně zobrazovaní[1].

Pochopitelně program je tvořen několika dalšími třídami. Předně dvě dokovací okna mají každé svoji třídu: `EditDockWidget` a `InfoDockWidget`. `EditDockWidget` se stará o pohodlné zadávání dat. `InfoDockWidget` je univerzálně naprogramován k pohodlnému zobrazování informací.

Třída `Printing` jak název vypovídá se stará o tisk. Třída `QuickSearchDockWidget` se stará o vyhledávání. Pak zde máme několik tříd pro jednotlivá dialogová okna: `ExistingSwap`, `OrdersPrint` a `Settings`.

Maličko samostatně stojí třída `Undo`, která zastřešuje správu undo/redo a vytváření swapovacího souboru. Nemusí být zjevné, že také provádí veškeré změny dat. Blíže je toto vysvětleno v jejím popisu.

TODO: Třída `MainWindow` je skutečně pěkný tlouštík. Bylo by docela pěkné ji nějak zmenšit a některé části přesunout jinam. Třeba by mohlo být načítaní nastavení a jeho ukládání řešeno ve třídě `Settings`. Další věc je, jestli je vůbec dobrý nápad provádět vyhledávání v rámci třídy `MainWindow`.

TODO: Inicializace `MainWindow` je pěkně chaotická, chtělo by to do ní zanést řád. V minulosti například způsobovalo dost problémů a v současné podobě se mi opravdu nelíbí.

Struktura dat

Nejprve se podívejme na abstraktní realizaci dat. Data si lze představit jako dvě tabulky – tabulku dokladů a tabulku zakázek. Mezi nimi je relace typu `n..m`. V programu je to řešeno pomocí `QMap`, které v sobě obsahují další `QMap`u detail s čísly a částkami v druhé tabulce. Kvůli časovým důvodům jsem se rozhodl mít v paměti uloženu relaci obousměrně. To je z hlediska údržby složitější, ale uspoří se čas. Naopak na disku si ukládám jenom tabulku dokladů a tabulku zakázek z ní vytvořím. Uživatel totiž zakázky přímo nikde netvoří, vznikají a automaticky zanikají při tvorbě dokladů. K těmto dvěma tabulkám máme navíc strukturu `Data_Company`, ve které jsou informace o firmě.

Protože se s `QMap`ami špatně pracuje, například se špatně prochází, máme ještě navíc pole `QStringList` s čísly dokladů a čísly zakázek.

Architektura je potom typická pro Qtečka. Máme model, ve kterém jsou uložena data (v našem případě `DataModel`). Máme pohled, který data zobrazuje (v našem případě centrální Widget `MainWindow`, třída `QTableView`). Tento milostný trojúhelník uzavírá delegate, který reprezentuje jedno políčko tabulky. Ten je v našem případě `DataDelegate` a umožňuje nám jednu novou vlastnost, vysílá signál při změně textu.

TODO: Chtělo by to maličko přepsat ukládání a načítání dat. Ty funkce jsou napsané hrozně moc složitě. Vůbec celý `DataModel` by to chtělo trochu zkrátit. Třeba by nebylo špatné rozdělit dlouhé funkce jako `data`.

---

[1] To je ještě zajištěno pomocí standardní Qt třídy `QTableView`, přesně ve stylu architektury model/pohled.

Třídy a koncepty v detailech

V této kapitole se podíváme trochu detailněji na celý program. Čtenář by už měl mít základní přehled o jednotlivých komponentách.

MainWindow

Tato třída se stará o věci grafického rozhraní. Při inicializaci vytvoří ostatní třídy a inicializuje je. Poté vytvoří menu a toolbar a nastaví `QAction`.

Datová tabulka zobrazuje různé druhy dat a právě `MainWindow` je přepíná. Máme tyto tabulky: `TABLE_ORDERS`, `TABLE_RECORDS`, `TABLE_ODETAIL` (detail zakázky), `TABLE_NEWRECORD` a `TABLE_COMPINFO` (informace o firmě).

Tato třída také zastřešuje vyhledávání v textu a načítání nastavení pomocí `QSettings`.

TODO: A je v ní pěkný bordel!! Chtělo by ji to dost projít a maličko překopat. Některá místa by bylo prima přepsat.

DataModel

Tato třída obsahuje gettery na data všeho druhu, pomocí ní přidáváme nové doklady a mažeme je. Editace funguje pomocí kombinace obojího. Dále se stará o ukládání a načítání z disku. A má spousty pomocných funkcí (počítáni zbytku při rozúčtovávání, ...).

EditDockWidget

Třída se stará o zadávání informací o novém dokladu a editaci stávajících. Samotné čtení a vkládání dat je realizováno `MainWindow`. Z vnějšku jsou zajímavé funkce `NewRecord()`, `EditRecord()` a `Clean`. Ty nastavují jednotlivé módy toho widgetu.

InfoDockWidget

Jednoduchá třída, jejíž zdrojov- kód je přímočarý. Základní funkce, které ji využívají, jsou tyto: `setText()`, `setTemporaryText()`, `setStrings()` a `setPrefix()`. S jejich pomocí třída zobrazuje informace, stará se o automatické doplňování textu a podobné věci. Každé automatické doplňování má svoje jednoznačné ID v rámci programu. Ostatní funkce jsou poměrně dobře opoznámkovány ve zdrojovém kódu.

Undo

Třída `Undo` spravuje tzv. `UndoQueue`, což je obousměrný spojový seznam. Jeho jednotlivé uzly jsou informace o provedených operacích. `UndoQueue` je abstraktní základní třída bez dat a implementace metod. Každý z uzlů má tři základní funkce a další dvě informační. Základní funkce jsou `undo()`, které odvolá akci (provede opačnou), `redo()`, které provede akci a `read()`, které načte informace ze swap-souboru. Informační funkce jsou `getDescription()` s základním popisem a `getInfo()` s podrobnými informacemi.

Hlavní třída `Undo` je pak řízena pomocí funkcí `undo()`, `redo()` a `addUndoStep()`, na což vrací odpovědi zase `MainWindow`. Také provádí samotné akce, například při `addUndoStep()` se provede `redo()` této akce. Dále se stará o swapfile, zapisuje do něj informace, v případě pádu čte a zobrazuje okno `ExistingSwap`.

TODO: Odebrat třídu `UndoStartQueue`, protože je nadbytečná.

Printing

Tisk má dva módy a stejně tak třída `Printing`. Proto nejprve pomocí `setMode()` zvolíme mód a případně nahrajeme seznam zakázek pomocí `setOrderList()`. O tisk se postará funkce `startPrinting`. Ta nejprve zobrazí dialog pro tisk a poté vytvoří tiskové sestavy a vykreslí je. Pro pozicování pomocí řádek používáme procentuální pozicovací funkce. Bližší informace přímo v souboru.

Při tisku jednotlivých zakázek je jednodušší samotný tisk simulovat a tak spočítat, kam vyjdou jednotlivé doklady a kolik to bude mít všechno stránek. `Printing` si data přímo načítá z `DataModelu`.

Funkce pro vypisování částek

V souboru `currency.cpp` jsou definovány funkce pro vypisování částek. Dostávají `double` a další parametry a vracejí `QString` s naformátovanou částkou, třeba i s HTML barvičkami, bez znaménka, se znaménkem, ...

TODO: Možná by nebylo špatné ty funkce nějak sdružit do třídy, lépe řečeno některé metody by mohli zůstat skryté. Ta třída asi ne, leda by mělo smysl do ní vnořit i data a místo částek v `doublech` je mít v `Currency` třeba. Co takhle jen přesunout metody z `currency.h` do `currency.cpp`??

Třídy dialogových oken

Ty považuji za natolik jednoduché, že se mi s nimi nechce rozepisovat. Víceméně mechanicky implementují různá klikátka a podobné věci. Vlastní logiku pak stejně většinou neprovádějí.