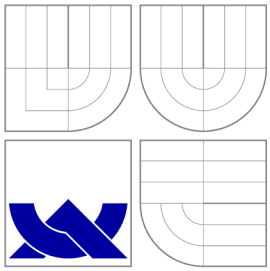
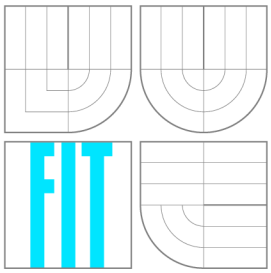


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

GUI KNIHOVNA PRO J2ME

GUI LIBRARY FOR J2ME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAROSLAV STROUHAL
AUTHOR

VEDOUCÍ PRÁCE — ING. ALEŠ LÁNÍK
SUPERVISOR

BRNO 2009

Abstrakt

Cílem této bakalářské práce je vytvořit nový framework pro platformu J2ME a otestovat jeho použitelnost na demonstrační aplikaci. Stručně je charakterizována samotná oblast J2ME, možnosti vývoje aplikací pomocí stávajících frameworků, popis nově vzniklého frameworku a popis demonstrační aplikace.

Abstract

Aim of this bachelor thesis is create new framework for platform J2ME and test it's usability in demonstrational application. Shortly is characterized alone area J2ME, potentials in development application via current frameworks, description of new arisen framework and description of demonstrational application.

Klíčová slova

Java, J2ME, GUI, Framework, Grafika.

Keywords

Java, J2ME, GUI, Framework, Graphics.

Citace

Jaroslav Strouhal: GUI knihovna pro J2ME, bakalářská práce, Brno, FIT VUT v Brně, 2009

GUI knihovna pro J2ME

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Aleše Láníka.

.....
Jaroslav Strouhal
19. května 2009

Poděkování

Chtěl bych poděkovat panu Ing. Aleši Láníkovi za odborné vedení a cenné rady při řešení bakalářské práce.

Obsah

Kapitola 1

Úvod

V dnešní době vlastní velké procento lidí nějaký typ mobilního zařízení. Takováto zařízení mají určitá omezení co se týče výkonu i paměti, nicméně jejich masivní nasazení na trhu představuje zajímavé odvětví pro vývoj nových aplikací. Jednotlivá zařízení se od sebe značně odlišují.

Existují různé platformy na kterých lze programovat aplikace pro takováto zařízení. Aplikace je možné programovat přímo pro jedno konkrétní zařízení, ale takovýto přístup je nevýhodný z hlediska použitelnosti v jiných zařízeních. Daná aplikace se pak musí přizpůsobit novému zařízení, ne vždy je to snadné, či dokonce možné.

Jiný přístup jak programovat aplikace je využití platformy, která není závislá na kontrétním zařízení. Nejznámější a hojně využívanou nezávislou platformou je *J2ME*, což je odvětví programovacího jazyka Java.

J2ME přináší spoustu výhod, ale také nevýhod, které jsou popsány v kapitole 3. Podpora dané platformy je u zařízení velmi dobrá. Výrobci mobilních zařízení s touto platformou počítají při distribuci nových produktů.

Nejznámějším odvětvím pro mobilní zařízení jsou Java hry. Exituje i velké množství různých organizérů, kalkulaček, jazykových překladačů a pod. Takovéto aplikace nemohou sice plně nahradit desktopové aplikace, ale přesto mají výhodu v tom, že si je uživatel může jednoduše přenášet s mobilním zařízením.

Opakující se problémy zdržují vývoj aplikací, proto vznikají různé *frameworky* pro urychlení, popř. zefektivnění realizace nové aplikace. Takovéto frameworky existují i pro *J2ME* a některé z nich jsou popsány v kapitole 4.

Tato bakalářská práce má za cíl popsat platformu *J2ME*, využití existujících *frameworků*, vytvoření a popsání vlastního *frameworku* na platformě *J2ME* a výsledný *framework* otestovat na demonstrační aplikaci.

Kapitola 2

Historie jazyka Java

Společnost Sun Microsystems chtěla v roce 1990 vytvořit nový programovací jazyk určený pro spotřební elektroniku. Spotřební elektronika má různé procesory a omezenou pamět. Požadavky na jazyk tudíž byly: přenositelnost, nenáročnost, bezpečnost a jednoduchost. Na základě těchto požadavků v roce 1991 vytvořil tzv. "Green team"pod vedením Jamese Goslinga objektově orientovaný programovací jazyk *Oak*. [8] [9] [6].

- Jazyk Oak,
- jazyk Java,
- J2SE, J2EE, J2ME.

2.1 Jazyk Oak

Oak vychází z jazyka C++, který i přes své výhody trpí nedostatky, jako jsou programátorské chyby (překročení rozsahu indexu pole, přístup mimo přidělenou paměť a pod.). Tyto chyby pak vedou na nespolehlivý software. Oak se snaží chybám předcházet např. odstraněním ukazatelů do paměti a správy paměti řízené programátorem. Některé chyby z C++ jsou ošetřovány již během překladu. Vzniká tak bezpečnější software, který je potřebný pro tvorbu aplikací pro spotřební elektroniku.

Prvním prototypem, na který byl jazyk nasazen, byl domácí ovladač *Star7*. Zařízení bylo podobné dnešním PDA — obsahovalo: LCD dotykový displej, bezdrátovou síť a infraport. Vyžití tohoto ovladače spočívalo v možnosti spouštět aplikace napsané v jazyce Oak a ovládání spotřebičů na dálku.

U spotřebitelů se neobjevila potřebná poptávka a projekt skončil nezdarem. V této době se na trhu objevila poptávka po novém druhu softwaru, určném pro internetové prohlížeče. Tato poptávka umožnila kolem roku 1993 vzniku tzv. appletů programovatelných v jazyce Oak. Prvním prohlížečem, který tyto applety využíval, se stal v roce 1995 Netscape Navigator od společnosti Netscape.

2.2 Jazyk Java

Oak byl v roce 1995 přejmenován na *Java*. Java byla poprvé představena na konferenci SunWorld 1995. Program v jazyce Java je překládán do bytecodu. Bytecode je interpretovaný pomocí Java Virtual Machine, díky které může program fungovat na různých platformách.

Z důvodu zvýšení rychlosti se používá *JIT kompilace*, která přeloží bytecode do nativního kódu.

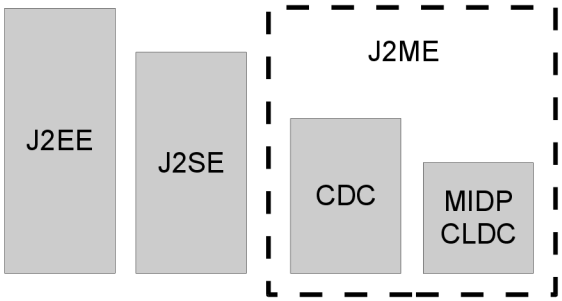
JIT kompilace znamená "Just in time", resp. neprovádí se naráz celá kompilace bytecodu do nativního kódu, ale kompiluje se jen ta část kódu, která je potřeba. Při dalším použití takto přeloženého kódu se již kód znovu nekompiluje. Tento přístup značně urychluje startování aplikace.

Přenositelnost mezi platformami vzbudila zájem vývojářů bežných aplikací. Mezi výhody interpretovaného přístupu patří: jedna aplikace může fungovat na různých operačních systémech bez nutnosti překladu, čímž se značně snížily náklady na vývoj aplikací. Nevýhodou byla absence ovládacích prvků, na které byli programátoři zvyklí z operačních systémů.

Kvůli zachování nenáročnosti pro "malé zařízení"typu domácí spotřebič, a zároveň aby bylo možno tvořit plnohodnotné Desktopové aplikace, došlo na rozdělení Javy na 3 platformy: J2SE, J2EE a J2ME. K tomuto rozdělení došlo před vydáním Javy 2.

2.3 J2SE, J2EE, J2ME

Následující obrázek srovnává jednotlivé platformy z hlediska podpory různých technologií.



Obrázek 2.1: Srovnání platforem Java. [3]

Java Standard Edition (J2SE)

Platforma pro tvorbu desktopových aplikací na klientské straně. Obsahuje balíky umožňující tvorbu grafického uživatelského rozhraní (AWT, Swing). Je možné používat standardní balíky, nebo komerční balíky naprogramované třetí stranou.

Java Enterprise Edition (J2EE)

Platforma pro tvorbu robustních projektů na straně serveru. Nabízí možnost autentizace a autorizace přístupu, připojení k databázi, komunikaci s klienty atd. Integruje nové technologie jako JSP, nebo servlety.

Java Micro Edition (J2ME)

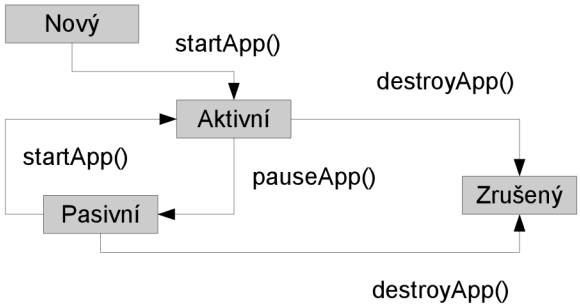
Platforma pro tvorbu aplikací pro zařízení s omezeným výkonem a pamětí (PDA, mobilní telefony, atd.). V roce 1998 společnost Sun Microsystems obnovila vývoj v této oblasti a v roce 1999 představila novou platformu pro vytváření aplikací pro vestavěné spotřebiče.

Kapitola 3

Seznámení s J2ME

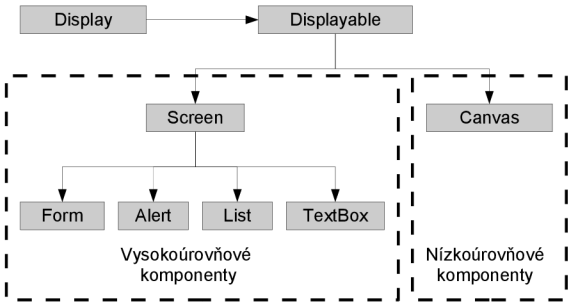
J2ME je jednou ze tří platforem javy. Ze všech ostatních platforem je nejchudší co se týče implementace. Chybí zde velký počet tříd obsažených např. v *J2SE*. Je to z důvodu, že platforma *J2ME* je určena pro malé zařízení typu telefon. Takováto zařízení mají obvykle značně omezený výkon procesoru, velikost pevné paměti i velikost operační paměti proti bežným počítačům.

Aplikace řídí aplikační manažer, který má na starosti stavy aplikace a překreslování grafiky. Následující obrázek reprezentuje stavový diagram aplikace. [4]



Obrázek 3.1: Životní cyklus aplikace.

Platforma obsahuje nízkoúrovňové a vysokoúrovňové komponenty. Následující obrázek popisuje jejich rozdělení.



Obrázek 3.2: Rozdělení základních komponent J2ME. [8]

- Vysokoúrovňové komponenty,
- nízkoúrovňové komponenty,
- Record Management System,
- výhody,
- nevýhody.

3.1 Vysokoúrovňové komponenty

J2ME obsahuje paletu základních komponent, které nejdou vzhledově modifkovat, ale lze díky nim sestavit jednoduché formulářové aplikace. Mají omezené použití a jejich funkcionalita nemůže být dále rozšiřována.

3.1.1 Alert

Slouží pro vytvoření vyskakovacího okna se zprávou. Účel této komponety je upozornit uživatele na chybu, nebo aby rozhodl o následující akci a pod. Může nabývat typu:

- **ALARM** — Informuje uživatele o dokončení některé započaté úlohy.
- **CONFIRMATION** — Uživatel je vyzván aby potvrdil, nebo zamítl danou výzvu.
- **ERROR** — Chybové hlášení.
- **INFO** — Zpráva pro uživatele má čistě informativní charakter.
- **WARNING** — Varování pro uživatele, že např. nějaká operace nebyla úspěšná.

3.1.2 Form

Vytváří formulář do kterého lze přidávat další komponenty, nebo je mazat. Komponentám lze nastavit preferovanou velikost, nastavit *Layout* pro vykreslení a namapovat řídící příkaz, který bude komponenta generovat. Komponenty jsou typu *Item*.

- **ChoiceGroup** — Umožňuje uživateli si vybrat ze skupiny možností. Možnosti jsou buď jen v textové podobě, nebo mohou být doprovázeny obrázkem.
- **CustomItem** — Jediná z komponent u které lze úspěšně změnit vzhled a funkcionalitu.
- **DateField** — Zobrazuje kalendář, ze kterého si uživatel může vybrat datum. Datum lze i programově přednastavit.
- **Gauge** — Druh ukazatele jakoby graf. Je u něj možné nastavit maximální a aktuální hodnotu. Komponenta tyto hodnoty převede do grafické reprezentace.
- **ImageItem** — Obrázek, který se bude zobrazovat ve formuláři. Lze nastavit zástupný řetězec, kdyby se obrázek nezobrazil.
- **Spacer** — Vytváří volné místo ve formuláři. Neposílá žádné zprávy, ani nic nezobrazuje na displeji. Slouží čistě jako oddělovač ostatních komponent.

- **TextField** — Jednořádkové textové políčko. Mezi jeho nastavení patří: maximální počet znaků, filtrování znaků, neboli jaká sekvence znaků může být do něj zadána – např. url, telefonní číslo, jen čísla .., a typ zobrazení např. heslo, že se místo napsaných znaků zobrazují zástupné symboly (např. hvězdičky).
- **StringItem** — Zobrazuje text na displeji, uživatel jej nemůže editovat. Editovat jeho text lze pouze programově.

3.1.3 List

List má velmi podobnou funkcionalitu jako *ChoiceGroup*. Hlavní rozdíl je v tom, že *ChoiceGroup* je komponenta určená pro přidání do komponenty *Form* a komponenta *List* zabírá celý displej.

3.1.4 TextBox

Víceřádkové textové políčko, které zabírá celý displej. Filtrování uživatelského vstupu a maximální délku zadaného textu je možné nastavit jako u *TextField*.

3.2 Nízkoúrovňové komponenty

Jedinou nízkoúrovňovou komponentou, která je určena pro celou obrazovku je *Canvas*. *Canvas* zachytává uživatelské vtupy z klávesnice, nebo dotykové obrazovky. Problémem uživatelského vstupu z klávesnice je fakt, že lze správně poznat jen tlačítka obsažené v tzv. *game actions*, numerické klávesy, mřížku a hvězdičku. Ostatní klávesy nejsou standardizované a každý výrobce zařízení je namapoval jiným způsobem.

Game actions jsou určené k vytváření her, ale samozřejmě mohou být využitelné i k jíným účelům. Mezi tyto klávesy patří šipky, tlačítko ok a herní akce A, B, C a D. Ani game actions nejsou úplně dokonalé, protože šipky a tlačítko ok může být namapováno pod numerické klávesy. Rozdělení kláves pro různá zařízení se dá vyřešit specializovaným překladem pro dané zařízení. Toto řešení však není dokonalé z důvodu, že daná aplikace již není multiplatformová, což odporuje základní myšlence jazyka Java.

Dalším aspektem je grafika. Zařízení mají různé parametry displejů co se týče velikosti, podpory barev, nebo propustnosti alfa kanálu. Řešení může spočívat taktéž ve specializovaném překladu jako u vstupů z klávesnice, nebo ošetření daných vlastností uvnitř každé aplikace. Informace o displeji shromažďuje třída *Display*.

Vlastní vykreslování je zajištěno *aplikačním manažerem* voláním metody *paint* a získáním *grafického kontextu* při potřebě překreslit část, nebo celý displej. Odkaz na *grafický kontext*, který může vykreslovat na obrazovku, je možné získat jen od aplikačního manažeru.

Voláním metody *repaint* je aplikačnímu manažeru poslána žádost o překreslení části obrazovky, nebo celé komponenty. Překreslování může být v určitém typu aplikací, jako jsou hry, klíčové a lze si jej vynutit okamžitě voláním metody *serviceRepaints*. Při tomto typu překreslení aplikační manažer nehlídá vykreslování a může dojít k *deadlocku* — seknutí aplikace.

3.3 Record Management System

Dále jen RMS. Prostor pro ukládání *perzistentních dat*.

Perzistentních data jsou taková data, která přetrvávají i po vypnutí programu, nebo napájení. Typickým příkladem takovýchto dat jsou relační databáze. [10]

Jedná se velmi jednoduchou databázi dat, obvykle má omezenou paměť v závislosti na daném zařízení. Data se ukládají do tabulek tzv. *Store*. Záznamy v tabulkách mají číselný identifikátor, začínající od 1, a bytové pole reprezentující data.

Tabulky je možné vytvářet, mazat a nastavovat jim přístupové právo, resp. jesli smí k tabulce přistupovat jen aplikace, která tabulku vytvořila, nebo aby byl přístup neomezen. Jednotlivé záznamy je možné přidávat, číst, mazat, filtrovat a řadit. [5]

K filtrování a řazení slouží rozhraní *RecordComparator* a *RecordFilter*.

3.4 Výhody

Sestavení jednoduchého formulářového programu je poměrně jednoduchá záležitost, každý ovládací prvek je sestaven pro jednu obrazovku displeje.

Síťová spojení realizuje třída *Connector*. Platforma přímo podporuje http protokol díky rozhraní *HttpConnection*, popř. lze komunikovat pomocí socketů díky rozhraní *SocketConnection*.

Hlavní výhodou platformy je její bezpečnost. Aplikace běží v tzv. sandboxu, neboli na svém vlastním pískovišti, které nemůže opustit. Pokud aplikace potřebuje přístup k některým potencionálně nebezpečným službám (např. síťovým), tak tuto operaci musí uživatel explicitně povolit. Po podepsání vytvořené aplikace již nemusí uživatel povolovat nebezpečné operace.

3.5 Nevýhody

CLDC 1.0 nepodporuje operace s plovoucí desetinnou čárkou, takovéto operace je nutné ošetřit vzlášť, nebo použít některý z externích balíků. Tyto operace jsou podporovány až od CLDC 1.1. Externí balíky mohou být komerční, nebo volně použitelné podobně jako v *J2SE*. Nelze modifikovat vzhled vysokoúrovňových komponent, ani je vzájemně do sebe vkládat - je možné vkládat pouze podkomponenty do komponenty *Form*, nebo elementy do *List*. Problémy způsobují nestandardní klávesy, popř. celé zařízení.

Kapitola 4

Existující frameworky pro J2ME

- Co je framework,
- J2ME Polish,
- OpenBaseMovil,
- LWUIT.

4.1 Co je to framework?

Framework je softwarová struktura, která pomáhá při nárvhu a vývoji aplikací. Typicky obsahuje knihovny, pomocné aplikace, nebo návrhové vzory.

Smyslem frameworku je přebrat typické problémy dané oblasti, kterou se zabývá, a tím zjednodušit vývoj nových aplikací.

Využití frameworku urychluje vývoj nové aplikace, ale nový framework je nutno nejprve nastudovat. Opakovaným využitím stejného frameworku se dosahuje kýženého rychlejšího vývoje. [13]

4.2 J2ME Polish

Jedná se o velmi rozsáhlý komerční framework od společnosti ENOUGH SOFTWARE, který obsahuje velké množství ovládacích komponent, podporuje perzistenci dat či lokalizaci. Jeho použití pro nekomerční účely je možné po zaregistrování na domovských stránkách a souhlasu s podmínkami užívání. Nejednotnost mobilních zařízení řeší vytvářením specializovaných verzí pro každý typ zařízení. [7]

Komponent je v J2ME Polish velké množstí, takže výčet komponent není úplný, ale jen ukázkový.

Celoobrazovkové komponenty

Daným komponentám je možno definovat nadpis, pozadí, menu a posuvník.

- **Alert**: Vyskakovací hláška.
- **FilteredList**: Zobrazuje informace na obrazovce, které je možno vybrat. Přidáváním znaků do textového políčka se informace filtrují.

- **Form**: Formulář do kterého je možné vložit další prvky.
- **TabbedForm**: Kontejner umožňující definovat možnosti výběru a pod každou možnost přiřadit prvek který se má zobrazit.
- **TextBox**: Textové políčko, které při psaní zobrazuje nápovědu.

Komponenty

- **HtmlBrowser**: Jednoduchý webový prohlížeč.
- **ClockItem**: Zobrazuje aktuální čas.
- **ImageItem**: Zobrazuje obrázek, nebo alternativní text.
- **MessageItem**: Zobrazuje 2 různé texty a případně obrázek.

Vlastní komponenty

Nové komponenty lze vytvořit rozšířením komponenty *CustomItem* a přidáním požadované funkčnosti.

Stylování

Stylování komponentů je přímo odvozeno od *CSS* modelu. Vzhled je ukládán do textového souboru. Každé nastavené zařízení má vlastní textvový soubor se vzhledem. Tento přístup má výhodu např. v tom, že zařízením s větším displejem je možné nadefinovat větší obrázek na pozadí, než zařízením s menším.

Lokalizace

Texty obsažené v aplikaci lze definovat ve zdrojích. Jeden soubor obsahuje právě jeden jazyk. V nastavení je nutné zvolit jaký jazyk se má použít. Programově pak stačí načíst si zvolený řetězec pomocí třídy *Locale* a čísla řetězce.

Perzistence

Framework podporuje perzistenci dat. Ukládat a načítat lze objekty, nebo jejich vektory.

Vzálené volání metod

Podporuje síťovou komunikaci se serverem. Obsahuje klientskou část pro mobilní zařízení a serverou část, díky tomu je komunikace značně ulehčena. Podmínkou je ovšem server založený na platformě *J2EE*.

Nevýhody

Jedná se o komerční projekt, takže je nutné si pro další komerční využití zakoupit licenci.

Shrnutí

Prozatím je tento projekt nejrozsáhlejším frameworkem. Nabízí spoustu možností a ulehčení. Projekt J2ME Polish je pořád upravován a vylepšován, takže i do budoucna má dobré vyhlídky.

4.3 OpenBaseMovil

OpenBaseMovil je framework s podporou databáze od společnosti Elondra. [1]

Databáze

Databáze nepodporuje sql dotazy. Přístup je realizován přes *JDBC ovladač*. Ukládá se do úložiště *RMS*. Pokud jsou data větší než maximální velikost úložiště, tak se data automaticky komprimují.

Java Database Connectivity (známé spíše jako *JDBC*) je *API* pro programátory v programovacím jazyku Java, které definuje jednotné rozhraní pro přístup k relačním databázím. Pro přístup ke konkrétnímu databázovému serveru je potřeba JDBC ovladač, který poskytuje tvůrce databázového serveru. [15]

Databáze podporuje vytváření a mazání tabulek, přidávání, mazání a filtrování záznamů, indexované vyhledávání, spojování tabulek atd. [2]

Souborový systém

Souborový systém je nasimulován do úložiště *RMS*. Tento přístup má výhodu, že aplikace nepotřebuje žádné povolení od uživatele a pod.

Přístup na tento souborový systém je realizován pomocí vstupních a výstupních proudů.

Serializace dat

Serializace je realizována vstupním a výstupním proudem. Do výstupního proudu jsou nahrány data daného objetku. Ze vstupního proudu se opět obnovuje objekt.

Logování

Umožňuje za běhu programu zapisovat poznámky k jednotlivím třídám do do úložiště *RMS*. Význam logování je převážně pro odladění aplikace, může pomoci vyřešit komplikované situace.

Matematika

Operace s plovoucí desetinnou čárkou nejsou v profilu CLDC 1.0. Framework tento problém řeší a umožňuje tyto operace využívat i tomto profilu.

MVC model

Model-view-controller (MVC) je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má minimální vliv na ostatní.[16]

Pohled, neboli view, je realizován implementací rozhraní View. Každý takovýto pohled musí mít definován unikátní identifikátor pro jeho snadnější nalezení a pohled, ze kterého byl vyvolán. Zjednodušuje se tím způsob jak se dostat na předchozí pohled — tzv. zpět.

Řídicí logika, neboli controller, je důležitou součástí architektury. Umožňuje reagovat na události v pohledu. Obsahuje rezervované základní čísla událostí systému, programátor si může definovat vlastní.

Lokalizace

Lokalizace pro jednotlivé jazyky je uložena ve zdrojích. Přístup k ní zprostředkovává třída ResourceManager. Při změně jazyka se automaticky posílá zpráva všem posluchačům.

Síťová komunikace

Framework umožňuje kryptovat a dekryptovat data. Výhodné zejména při posílání citlivých dat jako hesla a pod. Pro vyšší bezpečnost je možné zprávě přidat CRC a zabezpečit tak, aby data co byla zapsána se správně přečetla.

Nevýhody

Neobsahuje žádné komponenty, slouží jen jako návrhový vzor. Knihovna již není dále vyvíjena. Pro komerční využití je nutné si koupit licenci.

Shrnutí

Zajímavý návrhový vzor, který obsahuje použitelnou funkcionalitu, zejména databázi a oblast síťových služeb. I přes absenci konkrétní grafiky stojí za to tento framework alespoň vyzkoušet.

4.4 LWUIT

LWUIT [17] je poměrně obsáhlý a propracovaný framework od společnosti Sun Microsystems.

Podporuje průhlednost, dokonce i 3D zobrazení, dotykový displej, tlačítka různých zařízení. 3D zobrazení slouží pro různé vizuální efekty, jako otočení aktuálního obrazu na displeji na jiný, rotace tlačítek apod. Obsahuje různé vizuální efekty.

Podporovaná zařízení: Motorola, BlackBerry, Nokia S40, Sony Ericsson u ostatních není zaručena plná kompatibilita.

Komponenty

Obsahuje pestrou sadu základních komponent.

- **Component**: Základní komponenta zajišťuje grafické vykreslování na obrazovku a interakci s uživatelem.
- **Container**: Shromařďuje ostatní komponenty a umožňuje je pozicovat pomocí *layout manažerů*.
- **Form**: Formulář typu *Container*, který má nadpis, tělo, do kterého se vkládají ostatní komponenty, a menu.

- **Label**: Jednořádkové textové políčko i obrázek. Má možnost nastavit zarovnání textu.
- **Button**: Tlačítko na které je možno namapovat posluchače zpráv o stisknutí. Tlačítku může být nastaven text i obrázek.
- **RadioButton**: Typické výběrové tlačítko — určuje 1 z možností.
- **ButtonGroup**: Z *RadioButtonů*, které jsou do této skupiny přidávány, vytváří skupinu.
- **CheckBox**: Typické zaškrtávací tlačítko. Je možno namapovat posluchače zpráv o změně.
- **ComboBox**: Umožňuje výběr z více prvků. Je možno namapovat posluchače zpráv o změně.
- **TextArea**: Textové políčko u kterého lze nastavit počet řádků, maximum znaků, textový filtr a jesli jej je možno editovat.
- **TabbedPane**: Kontejner umožňující definovat možnosti výběru a pod každou možnost přiřadit prvek který se má zobrazit.

Vlastní komponenty

Nové komponenty lze vytvořit rozšířením základní komponenty *Component*, nebo jakékoli jiné komponenty, a přidáním požadované funkčnosti.

Stylování

Stylování komponentů je inspirováno *CSS* modelem. Nastylovat je možno jednotlivé komponenty, nebo skupiny komponent, díky tomu lze vytvářet i vlastní motivy aplikace.

Správa zdrojů

Knihovna obsahuje program na správu zdrojů, ve kterém je možno upravovat vzhled aplikace, obrázky, animace, fonty, lokalizaci a data.

Logy

Framework umožňuje vytvářet si logy, které jsou ukládány do *RMS*. Logy slouží k lepšímu monitorování a debugu aplikace. Existují 4 varianty logů: DEBUG, INFO, WARNING a ERROR.

Nevýhody

Knihovna je navržena spíše pro novější typy zařízení. Zabírá poměrně hodně systémových zdrojů zařízení. Vizuální efekty jsou na úkor rychlosti aplikace. 3D zobrazení je vytvářeno procesorem zařízení — může způsobit problémy při náročnějších aplikacích.

Shrnutí

I přes náročnost na hardwarové nároky se jedná o zájímavou knihovnu, zejména co se týče grafického designu.

Kapitola 5

Jsfw framework

- Výhody,
- vytvoření aplikace,
- komponenty,
- vlastní komponenty,
- výběr komponent,
- layout manažery,
- zprávy,
- grafika,
- stylovatelnost,
- uživatelské vstupy,
- časovač.

Cílem frameworku je vytvořit co nejpříjemnější prostředí z hlediska programátora i z hlediska uživatele. Jsfw využívá vlastností J2ME jako jsou uživateské vstupy a výstupy. Systém implementuje zasílání zpráv při událostech, časovač, vykreslování, dotykovou obrazovku a další vlastnosti. Uživatelské rozhraní Jsfw frameworku je založeno na komponentním programování.

Minimální nároky jsou: zařízení typu *CLDC 1.0* s podporou *MIDP 2.0*.

5.1 Výhody

Kompatibilita s různými zařízeními je zajištěna možností namapovat si čísla kláves. Lze využít dotykovou obrazovku, pokud ji cílové zařízení vlastní.

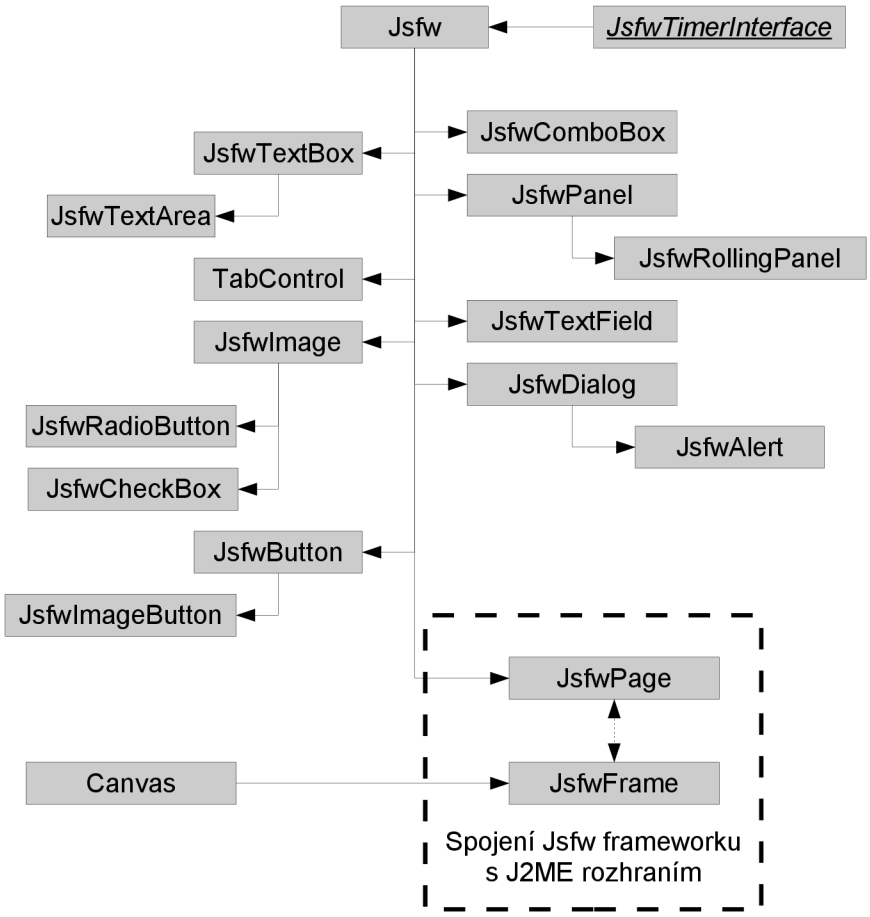
Jednotlivé komponenty je možno různě konfigurovat, vkládat je libovolně do sebe, pokud nemá tuto vlastnost komponenta zakázanou, nebo si vytvořit vlastní novou komponentu. Při vytváření nové aplikace lze zkombinovat standardní komponenty *J2ME* s komponentami *Jsfw*, ale jen tak, že v rámci jedné obrazovky (display) budou komponenty jednoho typu — buď *Jsfw*, nebo *J2ME*. Komponenty není nutné pozicovat ručně na obrazku, o nastavení pozice se může postarat jeden ze dvou připravených layout manažerů, nebo si vytvořit vlastní podle potřeby.

5.2 Vytvoření aplikace

Framework obsahuje 2 základní třídy nutné pro spuštění aplikace — *JsfwFrame* a *JsfwPage*. Objekt vytvořený podle třídy *JsfwFrame* musí být nastaven na obrazovku zařízení jako výchozí. Tento objekt vytváří rozhraní mezi vstupy a výstupy z *J2ME* a předává je *Jsfw* a přídávají se do něj jednotlivé stránky (*JsfwPage*), které reprezentují obrazovku zařízení a obsahují další komponenty.

5.3 Komponenty

Následující obrázek znázorňuje vztahy komponent ve frameworku a jakým způsobem framework komunikuje s *J2ME* rozhraním.



Obrázek 5.1: Rozdělení základních komponent Jsfw frameworku.

Jsfw

Základní komponenta, protože ostatní komponenty od ní dědí své vlastnosti. Je to jakási šablona, podle které jsou ostatní komponenty vytvořeny, ale sama nemá žádné speciální vlastnosti. Obsahuje a implementuje všechny základní vlastnosti jako např. změna barvy

pozadí, popředí atd.., je proto vhodná pro vytváření vlastních komponent. Zajišťuje vykreslování sebe a potomků přes metodu *paint*, vykreslování sebe je zajištěno metodou *paintComponent*, která je vhodná pro vytvoření vlastního vzhledu komponenty přetížením.

Obsahuje metody, které jsou volány po určité akci jinými objekty rozhraní. Předefinování těchto metod, je vhodné při vytváření nových komponent, není nutné používat všechny.

- **onAdded**: Voláno pokud je komponenta úspěšné přidána do jiné komponenty.
- **onChangeMinWidth**: Voláno při změně minimální šířky.
- **onChangeMinHeight**: Voláno při změně minimální výšky.
- **onChoose**: Voláno v případě, že komponenta nemůže získat focus a přesto byla vybrána.
- **onFocused**: Voláno po získání focusu.
- **onKeyPressed**: Voláno při stisku tlačítka (klávesy).
- **onLostFocus**: Voláno při ztrátě focusu.
- **onPressed**: Voláno při klepnutí na dotykový displej zařízení v případě, že dané zařízení takovou obrazovku vlastní.
- **onSoft1**: Voláno při stistku levého pomocného tlačítka.
- **onSoft2**: Voláno při stistku pravého pomocného tlačítka.
- **onUpdate**: Voláno časovačem. Označuje vypršení události, která je rozpoznatelná podle příchozího id.

JsfwFrame

Jako jediná komponenta frameworku není potomkem třídy *Jsfw*, ale *Canvas*. Zajišťuje vykreslování, uživatelské vstupy, správu *JsfwPage* (dále jen stránek) a vzájemnou komunikaci s aktuálně zobrazenou stránkou.

Stránek může tato komponta obsahovat libovolné množství, resp. do maximální velikosti operační paměti zařízení. Mezi vloženými stránkami lze volně přepínat, nebo je smazat. Přidávat další stránky do komponety je možné v jakýkoli okamžik běhu programu.

JsfwPage

JsfwPage, neboli stránka, vyplňuje celou obrazovku a zajišťuje komunikaci s *JsfwFrame*. Každá stránka definuje svůj vybraný prvek nezávisle na ostatních a zajišťuje změnu vybraného prvku.

Zasílá zprávy:

- **ON_ACTION**: Vybranému prvku byl přiřazen focus.
- **ON_CHANGE**: Vybraný prvek byl změněn.

JsfwPanel

Shromažďuje ostatní komponenty. Nemá žádné speciální vlastnosti. Může být použit jako kontejner s *layout manažerem* pro ostatní komponenty, nebo jako oddělovač komponent.

JsfwRollingPanel

Podobná funkcionalita jako *JsfwPanel*, dědí jeho vlastnosti. Rozdíl nastává pokud jsou do komponenty vloženy potomci přesahující její velikost. V takovém případě komponenta postupně posunuje své potomky ze strany na stranu, popř. nahoru a dolů, a zobrazuje tak všechen obsah, který by byl jinak skryt.

JsfwButton

Reprezentuje tlačítko, které je možné stisknout. Mění svou velikost podle zadaného textu tlačítka.

Zasílá zprávu:

- **ON_ACTION**: Akce je posílána v případě, že dojde k vybrání prvku.

JsfwIcon

Tlačítko s obrázkem. Posílá po stistknutí taktéž zprávu *ON_ACTION* jako *JsfwButton*. Svou velikost mění podle zadaného obrázku. Pokud je zadán text, tak jej vypisuje po najetí na komponentu pod ní.

Zasílá zprávu:

- **ON_ACTION**: Akce je posílána v případě, že dojde k vybrání prvku.

JsfwTextField

Jednořádkový textový nápis. Použitelný pro pomocné nápisy a podobně. Editovat lze jen programově, nikoli uživatelsky. Nedá se na něj najet, ani jej označit. Nemá nadefinované žádné speciální vlastnosti. Automaticky mění svou velikost podle textu, který zobrazuje.

JsfwTextBox

Jednořádkové textové políčko do kterého lze zapisovat text. Při označení prvku se zobrazuje blikající kurzor, znázorňující aktuální polohu v textu. Pohyb kurzoru je realizován šipkami vlevo a vpravo. V případě, že je zadaný text delší než velikost textového políčka, tak se po stránách objeví šipka na straně kde text přesahuje políčko.

Políčku lze nastavit vlastnost "nápověda", což znamená že pokud komponenta neobsahuje žádný text, tak zobrazuje text nápovědy. Nápověda zmizí v případě označení prvku, nebo přidání znaků do textového políčka.

Políčko lze využít i jako element pro zadávání hesla, díky změně zobrazování, kdy místo psaného textu se zobrazují zástupné znaky — např. hvězdičky a pod.

Zasílá zprávu:

- **ON_CHANGE**: Při změně textu.

JsfwTextArea

Víceřádkové textové políčko do kterého lze zapisovat text. Při označení prvku se zobrazuje blikající kurzor, znázorňující aktuální polohu v textu. Pohyb kurzoru je realizován šipkami vlevo, vpravo, nahoru a dolů. Zadávané slovo se zalomí na další řádek v případě, že šířka textu na řádku překročí šířku políčka. Pokud výška řádků překročí výšku komponenty, tak se nahoře, nebo dole, objeví pomocná šipka na straně kde text přesahuje políčko.

Zasílá zprávu:

- **ON_CHANGE**: Při změně textu.

JsfwComboBox

Prvek výběru z možností. Vložit do možností lze jakýkoli objekt typu *Jsfw*, takže možnosti dané komponenty nejsou nijak omezeny. Komponenta upravuje svou velikost podle přidaných možností tak, že šířku nastavuje podle největší šířky z možností a výšku podle největší výšky.

Zasílá zprávy:

- **ON_ACTION**: Akce je posílána v případě, že je komponenta vybrána a uživatel potvrdí vybraný prvek.
- **ON_CHANGE**: Při změně výběru z možností.

JsfwTabConstrol

Kontejner umožňující definovat možnosti výběru a pod každou možnost přiřadit prvek který se má zobrazit. Vložit do možností lze jakýkoli objekt typu *Jsfw*, takže možnosti dané komponenty nejsou nijak omezeny.

Zasílá zprávu:

- **ON_CHANGE**: Při změně vybraného prvku.

JsfwImage

Jednořádkový textový obrázek, nápis, nebo oboje. Použitelný pro pomocné nápisy, vykreslení obrázku, výběrový element do *ComboBoxu* a podobně. Editovat lze jen programově, nikoli uživatelsky. Nedá se na něj najet, ani jej označit. Nemá nadefinované žádné speciální vlastnosti. Automaticky mění svou velikost podle textu a obrázku, který zobrazuje. Text se zarovnává vpravo od obrázku.

JsfwCheckBox

Zaškrtávací tlačítko, které má stavy: zašktlé a nezašktlé. Obsahuje obrázek a text. Každý stav má svůj obrázek. Obrázky jsou definovány buď obecně pro všechny *JsfwCheckBoxy*, nebo pro konktrétní instanci.

Zasílá zprávu:

- **ON_CHANGE**: Při změně stavu.

JsfwRadioButton

Výběrové tlačítko, které má stavy: vybrané a nevybrané. Obsahuje obrázek a text. Každý stav má svůj obrázek. Obrázky jsou definovány buď obecně pro všechny *JsfwRadioButtony*, nebo pro konktrétní instanci. Grupují se podle rodiče do kterého jsou přidány.

Zasílá zprávu:

- **ON_CHANGE**: Při změně stavu.

5.4 Vlastní komponenty

Tvorba nové komponenty je poměrně snadná — stačí zdědit vlastnosti od třídy *Jsfw*. Takováto nová komponenta by ovšem neměla žádnou speciální funkčnost ani vzhled.

Nový vzhled se definuje předefinováním metody *onPaint*. Framework před samotným voláním zajistí nastavení vykreslovací oblasti, překreslení pozadí i hranice a nastavení barvy popředí.

Jsfw definuje metody, kterými oznamuje novou akci komponentě, např. získání focusu (viz strana 17). Funkčnost komponenty je možné ovlivnit např. časovačem, nebo implementováním rozhraní *JsfwCommandListener* a reakcí na získané zprávy.

5.5 Výběr komponent

Výběr komponenty realizuje stránka *JsfwPage*, nebo je možné programově vynutit vybrání komponenty. Stránka reaguje na šipky, které mění vybraný prvek, levé a pravé pomocné tlačítko, které slouží k vynořování a zanořování do komponent, a tlačítko ok, které dává prvku focus, nebo jej vybírá. Existují 2 fáze vyběru:

- **Select**: Uživatel je na komponentě najetý, ale ovládání je v režii stránky. Pokud je tato vlastnost zakázána, tak na komponentu nelze najet.
- **Focus**: Uživatel má komponentu vybranou k použití, ovládání je v režii komponenty. Pokud je tato vlastnost zakázána, tak nelze komponentu vybrat k použití.

5.6 Layout manažery

Při návrhu aplikace není vhodné pozicovat komponenty na konkrétní pozici v aplikaci. Využitím některého z připravených *layout manažerů*, popř. vytvořením vlastního, se pozicování komponent na obrazovce značně zjednodušuje. Takovýto manažer může být aplikován na kteroukoliv komponentu, která může mít potomky. Komponenty přidané do komponenty, která má nastavený *layout manažer*, se automaticky pozicují podle povahy manažeru. Přepočítávání pozice potomků probíhá automaticky v případě změny velikosti prvku s manažerem, nebo některého z jeho potomků.

Vlastní layout manažer

Základ nového manažeru je rozhraní *JsfwLayout*, které je nutno implementovat. Rozhraní obsahuje 2 metody:

- **rePositionChilds**: Metoda má za úkol provést vlastní přepočítání a přepozicování potomků zadaného prvku.
- **recalculateMinSize**: Přepočítává jakou by měl mít zadaný prvek nejmenší velikost. Návratová hodnota této metody se využívá k určení jesli se má volat metoda *rePositionChilds* či nikoli.

Připravené manažery

Oba připravené manažery mají možnost nastavit odsazení jednotlivých komponent a jejich horizontální i vertikální zarovnání. Tyto manažery mají statickou instanci s předdefinovanými parametry. Vytvořením nové instance umožňuje přenastavit jejich parametry. Neviditelné komponenty nejsou pozicovány.

- **JsfwHorizontalLayout**: Prvky pozicuje horizontálně vedle sebe. Automaticky nastavuje vhodnou šířku roztahovatelným prvkům.
- **JsfwVerticalLayout**: Prvky pozicuje vertikálně pod sebe. Automaticky nastavuje vhodnou výšku roztahovatelným prvkům.

5.7 Zprávy

V případě, že komponenta vlastní posluchače (rozhraní *JsfwCommandLister*), může posluchači zasílat zprávy. Posluchači přijde číslo zrávy, pomocný parametr a odkaz na prvek, který zprávu vyvolal. Číslo zprávy je takové číslo, které bylo namapováno na danou zprávu. Bez namapování zprávy se zpráva posluchači neposílá.

Zde je popsán obecný význam zpráv. U jednotlivých komponent je zmíněno pokud komponenta zasílá některou ze zpráv u jiné příležitosti než je obecný význam.

ON_ACTION

Speciální akce komponent. Obecně se tato zpráva neposílá, musí být extra volána. Komponenty, které danou akci posílají jsou popsány v odstavci Komponenty.

ON_ADDED

Komponenta je úspěšně přidána do jiné komponenty.

ON_CHANGE

Tato zpráva je opět speciálni jako *ON_ACTION*, tzn. že obecně se tato zpráva posílá, jen v případě, že ji komponenta podporuje. Obecně značí že se komponenta změnila, záleží na komponentě jakým způsobem.

ON_CHOOSED

Pokud komponenta nemůže získat focus i přesto byla vybrána. Od komponent, které mohou získat focus, není tato zpráva odesílána.

ON_FOCUSED

Informuje o tom, že daná komponenta získala focus a může přijímat uživatelské vstupy z klávesnice.

ON_LOSTFOCUS

Komponenta ztratila focus a již nepřijímá uživatelské vstupy z klávesnice.

ON_SELECTED

Pokud bylo najeto na komponentu. Tako akce se vyvolává jen když se změní výběr komponety na jinou.

ON_UNSELECTED

Daná komponenta přestala být vybraná.

ON_UPDATE

Vyvoláno časovačem, v pomocném parametru je identifikátor jednotlivých časovaných elementů.

5.8 Grafika

Grafika je realizována pomocí standartního vykreslování v J2ME, resp. třídou *Graphics*. Vykreslování začíná od komponenty *JsfwFrame*, která pošle grafický kontext aktuální stránce (*JsfwPage*). Stránka rozesílá grafický kontext všem potomků a její potomci zase jejich potomkům.

Jednotlivé komponenty si nastavují počátek vykreslování a *Clip*, což je velikost kreslící plochy. Komponenta si pak nastaví svůj font, překresluje se barva pozadí, hranice a nastavuje se barva na barvu popředí, až pak je volána metoda *paintComponent*, která realizuje samotný vzhled komponenty.

Na konec dojde na stránce k vykreslení vybraného prvku a celkovému vykreslení grafického kontextu na obrazovku.

Není nutné překreslovat vždy celý displej, ale jen danou oblast. Komponenty obsahují metodu repaint, která si vynutí překreslení jen v oblasti ve které se komponenta aktuálně nachází.

5.9 Stylovatelnost

Nastylování může probíhat na globální úrovni, která ovlivňuje všechny komponenty, nebo lokální u každého prvku vzlášť. Lokální styl prvku má přednost před globálním. Následuje popis globálních vlastností, které jsou zároveň i lokální a je možno je měnit.

- **Barva popředí**: Barva, která se nastavuje před vlastním vykreslováním komponenty. Většinou mění barvu textu.
- **Barva ohraničení**: Definuje jakou barvou se má vykreslovat ohraničení prvku, pokud prvek není vybraný, je vybraný, nebo má focus.
- **Barva pozadí**: Barva, kterou se překreslí podklad komponenty před samotným vykreslením.
- **Obrázek pozadí**: Obrázek, který se na pozadí komponenty vykreslí jako textura. Pokud je obrázek nastaven, tak je barva pozadí ignorována a vykresluje se pouze obrázek. Neplatí pokud je obrázek globální a barva pozadí nastavena lokálně, pak se vykresluje jen barva pozadí.

- **Styl pozadí**: Definuje jesli se má pozadí vykreslovat, či nikoli.
- **Font**: Nastavuje font písma. Ve frameworku nejsou podporovány vlastní fonty, pouze základní sada z třídy *Font*.

Následuje popis čistě lokálních vlastností.

- **Viditelnost**: Určuje jesli se má prvek vykreslovat či nikoli. Pokud není viditelný ignoruje se v připravených *layout manažerech*.
- **Nastavení textu**: Všem komponentám je možné nastavit text, ne všechny komponenty tento text využívají.
- **Povolení získání focusu**: Povoluje prvku získat focus, čímž přebírá od stránky řízení uživatelských vstupů.
- **Povolení označení**: Určuje jesli je možné prvek označit či nikoli. Pokud má tuto vlastnost zakázanou, tak uživatel na takovýto prvek nemůže najet.
- **Roztahování**: Definuje jesli se má prvek automaticky zvětšovat do maximální možné velikosti. Roztahování do šířky je nezávisle nastavitelné od roztahovaní do výšky.

5.10 Uživatelské vstupy

Uživatelské vstupy nejprve zachytává *JsfwFrame* a předává je aktivní stránce *JsfwPage*.

Vstup z klávesnice předává stránka aktuálně vybranému prvku, který má focus, voláním jeho metody *onKeyPressed*. Jesliže vybraný prvek nemá focus, tak zpracuje vstup sama stránka.

Vstup z dotykového displeje je komplikovanější. Stránka za prvné zkontroluje jesli bylo klepnuto na vybraný prvek, pokud ano předá mu informaci kam bylo klepnuto metodou *onPressed*. Pokud nebylo klepnuto na vybraný prvek, tak stánka vyhledává na který prvek bylo klepnuto a tomu nastaví focus, ale neinformuje prvek kam bylo klepnuto.

5.11 Časovač

Časovač je naimplementován nekonečnou smyčkou v novém vlákně, která kontroluje přidané *časovací objekty* — *JsfwTimerItem*. Tyto objekty obsahují informace o komponentě, která čeká na událost, jesli je časovač aktivní, počáteční čas a dobu v milisekundách, za kterou se zavolá událost. V případě že komponentě vyprčí čas a je aktivní, zavolá se u komponenty událost *onUpdate* s identifikátorem časovacího objektu.

Na jednu komponentu lze aktivovat libovolný počet časovacích objektů, které je možno rozlišovat podle identifikátoru. Ukazatel na takovýto objekt je možné získat přidáním nové události do časovače. Po přidání je časování neaktivní. Podle potřeby je možné časování spustit, pozastavit, nebo úplně zrušit. V posledním případě již bude daný časovací objekt nepoužitelný.

Kapitola 6

Aplikace: Plánovací kalendář

- Popis,
- gui,
- návrh aplikace,
- iCalendar.

6.1 Popis

Plánovací kalendář je demonstrační aplikací, která má ukázat možnosti frameworku. Aplikace zobrazuje svátek pro každý den a možňuje vyhledávat svátky podle jména. Dále umožňuje vložit novou událost do kalendáře, odstranit a filtrovat události. Aplikace může automaticky upozorňovat na následující událost v případě, že to událost vyžaduje.

Naplánované události je možno exportovat na ftp server, popř. importovat události z ftp serveru. Import a export je realizován pomocí standardního formátu *iCalendar*.

6.2 Gui

Grafické uživatelské rozhraní je rozděleno do 5-ti stránek.

Menu

Menu je první aktivní stránka po načtení aplikace. Na prvním řádku stránky je zobrazen svátek pro aktuální den. Pod svátkem se nacházejí 4 ikony odkazující se na stránku s kalendářem, stránku s importem a exportem a poslení ukončující aplikaci. Pod ikonami zobrazuje pod sebe vyfiltrované události aktuálního dne. Na tyto události je možné najet a vybráním je editovat. Jejich editace probíhá na stránce s událostí.

Import a export

Stránka pro import a export je totožná liší se jen v nadpisu. Uživatel musí zadat adresu serveru, přihlašovací jméno, heslo a adresu souboru na serveru. Stránka si pamatuje poslední úspěšné spojení a při dalším použití jej předvyplňuje. V případě neúspěšného spojení aplikace nahlásí chybu a dané spojení není zapamatováno.

Kalendář

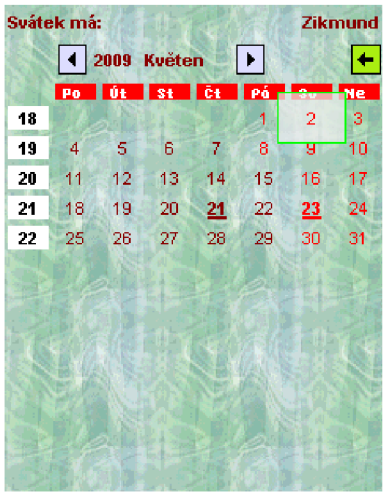
Po načtení kalendáře z menu je automaticky vybrán aktuální den, měsíc a rok. Na prvním řádku se zobrazuje název svátku ze dne na kterém je uživatel najetý. Pokud uživatel není najetý na žádném dnu, resp. je najetý na některém z ovládacích prvků, tak se nahoře místo svátku zobrazuje textové políčko s tlačítkem určené pro vyhledávání dne podle jména svátku. Toto vyhledání probíhá tak, že uživatel zadá do textové pole jméno svátku a pak klepne na tlačítko vyhledat. Pokud je svátek úspěšně vyhledán, tak se automaticky přemění měsíc a vybere se datum s daným svátkem. V opačném případě se nestane nic.

Období v kalendáři je možné měnit po rocích, nebo měsících. Změna po měsících má více variant. Uživatel může vybrat přímo měsíc z výběru měsíců, nebo šipkami se přepínat o jeden měsíc vpřed, nebo vzad, a poslední možností je při výběru dnů. Poslední možnost funguje tak, že když je vybraný den prvním dnem na daném řádku a uživatel stiskne šipku doleva, tak se změní měsíc o jeden zpět, nebo poslední den a uživatel stiskne šipku doprava, tak se změní měsíc o jeden vpřed. Období, kterým kalendář začíná je leden 2009, pod toto období se nelze dotat.

Dny, které mají namapovanou událost jsou zvýrazněny tučným a podtženým písmem. Volné dny jako státní svátky a víkend jsou zvýrazněny červenou barvou.

Ze stránky s kalendářem se lze vrátit na stránku s menu, nebo vybráním dne pokračovat na stránku s možnostmi dne.

Následuje ukázka stráky s kalendářem.



Obrázek 6.1: Ukázka stránky s kalendářem.

Možnosti dne

Na prvním řádku stránky je zobrazeno datum, tlačítko pro přidávání nové události, které se odkazuje na stránku s událostí, a tlačítko zpět na stránku s kalendářem. Zobrazuje seřazené události vybraného dne a umožňuje je filtrovat podle kategorie události. Na tyto události je možné najet a vybráním je editovat. Jejich editace probíhá na stránce s událostí. Pokud vybraný den nemá žádnou událost, tak se automaticky otevře stránka s přidáváním nové události. Tato stránka není nikdy zobrazena bez žádné události.

Událost

Stráka je učena pro editaci stávající události, nebo vytvoření nové. Hlavní rozdíl je tom, že vytvoření nové události má v dolní části stránky tlačítko storno na zrušení požadavku a editace má místo něj tlačítko smazat pro odstranění události. Jakmile se stránka vrací zpátky na stránku s možnostmi dne, je znovu zkontrolován den jestli vlastní alespoň jednu událost. V případě, že nevlastní (např. po smazání), tak se stránka automaticky přepne na stránku s kalendářem.

Uživatel může nadefinovat nadpis události, který se projevuje ve výpisech událostí, začátek a konec události, upozornění na událost a popis události, pokud je třeba.

6.3 Návrh aplikace

Logika aplikace, resp. přepínání mezi stránkami, je řešena na úrovni hlavní třídy *planningCalendar*. Jednotlivé stránky jsou definovány jako samostatná třída. Tento přístup není nutný, ale z důvodu přehlednosti je výhodnější. Každá stránka si svou vnitřní logiku řeší sama ve své třídě.

Aplikace demonstruje vytvoření nových komponent frameworku. Následuje výpis nových komponent.

- **CalendarItem**: Objevuje se na stránce s kalendářem. Reprezentuje jeden den. Nastavuje se podle zadaného svátku, kolekce událostí daného dne a podle toho jesli se jedná o víkend.
- **Event**: Objevuje se na stránkách s menu a možnostni dne. Reprezentuje jednu událost. Nastavuje se podle zadané události. Zobrazuje čas začátku události, ikonu kategorie a nadpis události. Nadpis zobrazuje vždy ostatní parametry jen podle události.

6.4 iCalendar

iCalendar je standard pro výměnu kalendářových dat, umožňuje uživatelům posílat žádosti k naplánování schůzky, nebo úkolu ostatním uživatelům prostřednictvím standardního emailu. [14]

Následuje výčet základních objektů a jejich implementace v aplikaci.

VCALENDAR

Základní objekt, který obsahuje ostatní. Mezi jeho vlastnosti patří definice časové zóny, verze a název produktu, kterým byl daný VCalendar vygenerován. Objekt má počátek a konec.

V aplikaci je implementován jako perzistentí objekt s jedinou statickou instancí. Zajišťuje vygenerování formátu iCalendar z dat a import z formátu zpět do dat.

Generování formátu probíhá procházením *RMS* objektů *VEVENT* a postupným voláním jejich metody *toString*. Výsledkem je vygenerovaný řetězec, který se následně pošle ftp serveru.

VTIMEZONE

Nastavuje časovou zónu. Určuje nejen časovou zónu, ale i změnu letního a zimního času, název časové zóny atd.

Tato vlastnost není implementována v aplikaci, resp. je nastavena staticky a je neměnná.

VEVENT

Základní prvek události. Objekt má počátek a konec. Má následující vlastnosti.

- **DTSTAMP**: Vytvoření události, vlastnost je typu *DateTime* a je vždy v utc tvaru.
- **DTSTART**: Začátek události, vlastnost je typu *DateTime*.
- **DTEND**: Konec události, vlastnost je typu *DateTime*.
- **TRANSP**: Má 2 stavy: TRANSPARENT, který nezabírá čas pro vyhledávání volného času, a OPAQUE, který naopak tento čas zabírá. Aplikace podporuje pouze TRANSPARENT, OPAQUE je bráno jako TRANSPARENT.
- **RRULE**: Opakování události. Viz strana 27.
- **SUMMARY**: Nadpis události.
- **DESCRIPTION**: Popis události.
- **VALARM**: Upozornění na událost. Viz strana 28.

Je implementován jako perzistentní objekt a většinu času se nachází pouze v *RMS*. Vyvoláván z úložiště je jen když je potřeba, například při výpisu událostí dne a pod.

Jako filtr událostí slouží třída *VEventFilter*, která filtruje události podle zadaného dne a vrací jejich kolekci. Umožňuje i filtrování podle upozorňení na událost.

DateTime

Vlastnot určuje datum, popřípadně čas. Jedná se o jednořádkovou vlastnost.

Je implementována jako objekt typu *SerializableData*, resp. že je daný objekt schopný se serializovat na pole bytů a zpětně deserializovat z pole bytů na objekt. Má 3 varianty.

- **UTC**: Uložen datum a čas ve formátu UTC.
- **GMT**: Uložen datum a čas ve formátu GMT. Je uvedena časová zóna.
- **NONE**: Je uveden pouze datum, bez času.

RRULE

Umožňuje vytvořit opakující se událost. Obsahuje různé možnosti pro opakování, např. poslední neděle v měsíci, každou druhou hodinu atd. Jedná se o jednořádkovou vlastnost.

Je implementována jako objekt typu *SerializableData*. Implementace dovoluje tento objekt načíst z formátu iCalendar i zapsat jej zpět, nicméně tato vlastnost není podporována filtrem událostí.

VALARM

Definuje časový interval ve kterém má aplikace upozornit na danou událost předem, nebo po jejím skončení. Objekt má počátek a konec.

Je implementována jako objekt typu *SerializableData*.

6.5 Perzistence dat

Aby bylo možné aplikaci reálně využívat, bylo nutné vyřešit problém persistence dat. Java SE má možnost tzv. serializace, kdy je objekt převeden na sekvenci bytů, popř. ze sekvence bytů lze získat zpět daný objekt. Jako persistence prvku se může využít např. uložení do souboru. Toto řešení není možno využít v J2ME, protože serializace není podporována a přístup k souborům je omezen.

Problém řeší balík *persistentData*, který obsahuje třídu *SerializableData*.

SerializableData

Rozšířením této třídy a nadefinováním abstraktních metod *writeObject* a *readObject* umožňuje serializovat objekt a znovu jej deserializovat. Abstraktním metodám jsou posílány vstupní a výstupní datové proudy, které slouží k uložení objektu a jeho opětovnému obnovení.

Persistent

Persistent je potomkem *SerializableData*. Abstraktní metody musejí být taktéž nadefinovány jako *SerializableData*.

Uložení objektu prohíhá tak, že je tento objekt nejprve serilizován na pole bytů a toto pole je pak uloženo do *RMS*. Načtení objektu probíhá tak, že je z *RMS* načteno pole bytů a to je deserializováno zpět na objekt.

PersistentService

Zajišťuje samotné ukládání do *RMS* a načítání z *RMS*. Ukládání automaticky pozná jesli jde o nový záznam, nebo o úpravu stávajícího.

Náčítání má dvě možnosti.

- **LoadCollection**: Vrací kolekci objektů zadaného typu, které odpovídají zadanému filtru *RecordFilter* a seřadí je podle zadaného komparátoru *RecordComparator*.
- **LoadFirst**: Vrací první záznam, který nalezne.

6.6 Ftp

FTP (anglicky File Transfer Protocol) je určen pro přenos souborů mezi počítači, na kterých mohou běžet rozdílné operační systémy (je platformně nezávislý). [12]

Charakteristika

FTP je jeden z nejstarších protokolů využívající porty TCP/21 a TCP/20. Port 21 slouží k řízení a jsou jím také přenášeny příkazy FTP. Port 20 slouží k vlastnímu přenosu dat, to platí jen u aktivního módu. Umožňuje procházet, přidávat, nebo mazat soubory na serveru. Přístup k serveru bývá omezen *autentizací* pomocí jména a hesla, nicméně se heslo přenáší v nekryptované podobě, tudíž není protokol bezpečný.

U procesu autentizace se jedná o ověření identity uživatele služeb nebo původce zprávy. Typicky např. uživatelské jméno a heslo. [11]

Aktivní a pasivní připojení

Připojení k FTP serveru je možné realizovat v aktivním nebo pasivním režimu.

- **Aktivní režim**: Server navazuje připojení pro přenos dat, klient naslouchá na portu TCP/20, na kterém jsou přenášena data. Problém zpravidla nastává v případě, kdy se klient připojuje z privátní sítě a jeho IP adresa je překládána (NAT).
- **Pasivní režim**: Klient navazuje připojení pro přenos dat, kterému při sestavování připojení poslal server svou IP adresu a TCP port, na kterém naslouchá.

Implementace

Ftp klient nemá implementovány všechny vlastnosti ftp protokolu, ale jen náležitosti k uložení a stažení souboru. Komunikaci se serverem zajišťuje třída *SimpleFtp*. Všechny potřebné údaje je nutné zadat při vytváření nové instance dané třídy. Třída pak vytvoří nové vlákno a zkusí provést zadanou akci (uložení, nebo stažení souboru). Jedná se o síťové připojení, takže uživatel musí tuto akci explicitně potvrdit.

Po úspěšném připojení k serveru odešle informace o uživateli, přepne se do pasivního režimu a zkusí poslat, nebo stáhnout požadovaný soubor. Na konci činnosti se odhlásí z ftp serveru a ukončí připojení.

Třída komunikuje přes rozhraní *SimpleFtpListener*. Toto rozhraní obsahuje 3 metody.

- **ftpUploadReady**: Získává jako parametr výstupní proud, do kterého se má uložit soubor. Pomocí tohoto proudu je pak soubor uložen na server.
- **ftpDownloadReady**: Získává jako parametr vstupní proud, který obsahuje výsledný soubor.
- **ftpMsgCode**: Získává chybové hlášky a informaci o dokončení přenosu.

Kapitola 7

Závěr

Seznámil jsem se s platformou J2ME, jaké má výhody, nevýhody a její možnosti při tvorbě uživatelského rozhraní. Nastudoval jsem existující frameworky pro danou platformu.

Vývoj vlastního frameworku probíhal řešením základních vlastností, jako jsou vnořování, stylovatelnost a vykreslování komponent, přes vytvoření vlastních komponent, až po přidání layout manažerů a časovače.

U demonstrační aplikace při tvorbě gui jsem vyžíval jen vlastního frameworku, aplikace neobsahuje žádné grafické komponenty z J2ME. Využil jsem vlastností frameworku, jako jsou časovač, předávání zpráv, nebo přepínání mezi obrazovkami.

Celý framework a demonstrační aplikace jsou zdokumentovány v elektronické podobě zdrojovými kódy a projektovými dokumentacemi na přiloženém CD.

Doporučení pro další vývoj

Pro další vývoj této práce bych doporučoval rozšíření základní palety komponet obsažených ve frameworku. Například vyskakovací hláška (*JsfwAlert*) je jen informativní chybí potvrzovací, nebo chybová.

Bylo by vhodné rozšířit i základní sadu layout manažerů, jsou k dispozici jen dva základní. V oblasti stylovatelnosti by bylo možné přidat více možností, jako například odsazení prvku, vnitřní odsazení, šířka ohraničení, apod., a rozšířit možnosti stylovnání z globální a lokální úrovně ještě na úroveň komponent, která bude ovlivňovat všechny prvky jedné komponenty, a případně na úroveň stránky, která by ovlivňovala komponenty obsažené v dané stránce.

Co se týče demonstrační aplikace chybí některé vlastnosti formátu *iCalendar*, jako jsou opakování děje. Některé části, které byly stvořeny pro demonstrační aplikaci, jako například perzistence dat a ftp klient, by bylo možné zahrnout přímo do frameworku.

Literatura

[1] Cerezo, N.: openbasemovil-core 2.0 Developers Guide [online]. http://switch.dl.sourceforge.net/sourceforge/openbasemovil/OpenBaseMovil-core-2.0-DeveloperGuide.pdf, 2007-11-04 [cit. 2009-05-17].

[2] Cerezo, N.: openbasemovil-db Developer Guide [online]. http://switch.dl.sourceforge.net/sourceforge/openbasemovil/OpenBaseMovil-db-DeveloperGuide-3.0.02.pdf, 2008-03-04 [cit. 2009-05-17].

[3] Day, B.: Developing Wireless Applications using the Java 2 Platform, Micro Edition [online]. http://developers.sun.com/mobility/getstart/articles/wirelessdev/wirelessdev.pdf.

[4] Lucie Bittnerová: J2ME v kostce - první midlet [online]. http://interval.cz/clanky/j2me-v-kostce-prvni-midlet/.

[5] Muchow, J.: J2ME 101, Part 3: Inside the Record Management System [online]. http://www.ibm.com/developerworks/library/j-j2me3/, 2003-12-12.

[6] Novotný, L.: Historie a vývoj jazyka Java [online]. http://www.fi.muni.cz/usr/jkucera/pv109/2003p/xnovotn8.htm, 2003 [cit. 2009-04-03].

[7] software, E.: J2ME Polish: Documentation [online]. http://www.j2mepolish.org/cms/leftsection/documentation.html.

[8] Topley, T.: *J2ME V KOSTCE, Pohotová referenční příručka*. GRADA, 2002, iSBN 80-247-0426-9.

[9] Volkmann, M.: Java - What's All The Excitement About? [online]. http://www.ociweb.com/javasig/knowledgebase/May1997/Java1Hr.pdf, 1997-05 [cit. 2009-04-03].

[10] Wikipedia: Persistence (computer science) — Wikipedia, The Free Encyclopedia [online]. http://en.wikipedia.org/wiki/Persistence_(computer_science), 2009.

[11] Wikipedie: Autentizace — Wikipedie: Otevřená encyklopedie [online]. http://cs.wikipedia.org/wiki/Autentizace, 2009.

[12] Wikipedie: File Transfer Protocol — Wikipedie: Otevřená encyklopedie [online]. http://cs.wikipedia.org/wiki/File_Transfer_Protocol, 2009.

[13] Wikipedie: Framework — Wikipedie: Otevřená encyklopedie [online]. http://cs.wikipedia.org/wiki/Framework, 2009.

[14] Wikipedie: ICalendar — Wikipedie: Otevřená encyklopedie [online]. http://cs.wikipedia.org/wiki/ICalendar, 2009.

[15] Wikipedie: Java Database Connectivity — Wikipedie: Otevřená encyklopedie [online]. http://cs.wikipedia.org/wiki/Java_Database_Connectivity, 2009.

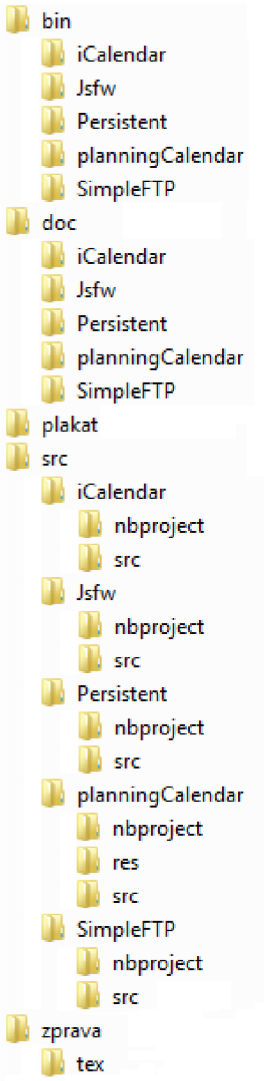
[16] Wikipedie: Model-view-controller — Wikipedie: Otevřená encyklopedie [online]. http://cs.wikipedia.org/wiki/Model-view-controller, 2009.

[17] WWW stránky: lwuit knihovna [online]. https://lwuit.dev.java.net/.

Příloha A

Obsah CD

CD obsahuje všechny náležitosti a to v následující adresářové struktuře.



Obrázek A.1: Adresářová struktura přiloženého CD.

Programová část je rozdělena v 5-ti projektech. Každý projekt je zastoupen v adresářích "bin", "src"a "doc".

V adresáři "bin"jsou spustitelné soubory jednotlivých projektů v adresáři s daným projektem. Dokumentace se nachází v adresáři "doc". Každý projekt má vlastní dokumentaci v adresáři s daným projektem.

Zdrojové soubory se nacházejí v adresáři "src". Opět má každý projekt vlastní zdrojové soubory v adresáři podle názvu projektu. Samotné zdrojové soubory jsou pak umístěny v podadresáři "src".

Bakalařská práce je uložena v adresáři "zprava". V jeho podadresáři "tex"je uložen zdrojový tvar písemné zprávy.

V adresáři "plakat"se nachází plakátek.