TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: B 2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Programování databázových aplikací v J2ME pro PDA

Programming of database applications in J2ME environment for PDAs

Bakalářská práce

Autor:	**Petr Vik**
Vedoucí BP práce:	Ing. Roman Špánek
Konzultant:	Ing. Pavel Pirkl

V Liberci 17. 5. 2007

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Katedra: Akademický rok: 2005/2006

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jméno a příjmení: Petr Vik

studijní program: B 2612 – Elektrotechnika a informatika

obor: 2612R011 – Elektronické informační a řídící systémy

Vedoucí katedry Vám ve smyslu zákona o vysokých školách č.111/1998 Sb. určuje tuto bakalářskou práci:

Název tématu:

Programování databázových aplikací v J2ME pro PDA

Zásady pro vypracování:

1. Student by se v první části měl věnovat možnostem jazyka Java ME
2. Zejména by se měl věnovat CDC konfiguraci a jejím možnostem
3. Vytvořit aplikaci, která by umožnila přímé spojení s libovolnou relační DB pomocí JDBC
4. Otestovat možnosti takové aplikace pro použití s více DB.

Rozsah grafických prací: dle potřeby dokumentace

Rozsah průvodní zprávy: cca 40 stran

Seznam odborné literatury:

[1] Internetové stránky Sun: http://java.sun.com/

[2] Bruce Eckel: Myslíme v jazyku Java, Grada Publishing, ISBN: 80-247-9010-6, 2001

[3] Qusay H. Mahmoud: Naučte se Java 2 Micro Edition, O'Reilly, 2003

[4] George Reese: Database Programming with JDBC and Java, Second Edition, O'Reilly, ISBN: 1565926161, 2000

Vedoucí bakalářské práce: Ing. Roman Špánek

Konzultant: Ing. Pavel Pirkl

Zadání bakalářské práce: **18.10.2006**

Termín odevzdání bakalářské práce: **19. 5. 2007**

L.S.

....

Vedoucí katedry Děkan

V Liberci dne

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé DP a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum:

Podpis:

Poděkování

Především děkuji svému vedoucímu bakalářské práce Ing. Romanu Špánkovi za jeho čas, cenné připomínky a odborné vedení při zpracování této práce.

Anotace

Práce se zabývá řešením způsobu komunikace mezi aplikací běžící na kapesním počítači PDA a databázovým serverem pomocí JDBC rozhraní. V úvodu je řečeno, jaké jsou dnešní možnosti J2ME a jaké možnosti přináší rozhraní JDBC. V dalších kapitolách je zpracováno jaká je podpora Javy na PDA a s jakým vývojem aplikací se můžeme setkat. Následně je popsáno nastavení SQL serveru a vytvořena aplikace v jazyku Java s konečným konstatováním, že se nepodařilo splnit zadání, protože nebyly dostupné dostatečné nástroje pro jeho realizaci. Závěr práce je věnován komerčnímu řešení společnosti Microsoft v podobě SQL Mobile, který je určen pro mobilní zařízení.

Klíčová slova

Java 2 ME, JDBC, CDC, PDA, databáze

Annotation

Work is dealt with solving way of communication between application which is running on pocket computer PDA and database server by force of JDBC interface. In introduction is told what the today's possibilities J2ME are and which possibilities bear an interface JDBC. In other chapters is worked up what support of Java is on PDA and which development of applications we can meet with there. Subsequently is described setting SQL server and created application in Java language with final enunciation that the task was not realized, because there weren't accessible competent tools for the realization. The end of work is devoted to commercial solution to the company Microsoft in form SQL Mobile that is intended for mobile arrangement.

Key words

Java 2 ME, JDBC, CDC, PDA, database

OBSAH

1. Úvod a cíl práce

Stále častěji se můžeme setkat s uživateli mobilních zařízení typu Palm nebo PocketPC využívající připojení k internetu, ať už pomocí mobilního operátora službou GPRS, nebo přímým spojením pomocí bezdrátové sítě WiFi. Naskýtá se otázka využití těchto zařízení jako přístupový bod k databázovému serveru. Pokud se nad tím zamyslíme, zjistíme že cílovou skupinou využívající takovouto aplikaci nebudou jen manažeři, ale i ti kteří mobilní zařízení opravdu potřebují.

Mobilní řešení databázové aplikace můžeme rozdělit do tří skupin. První je klasická databázová aplikace, která si stáhne nejnutnější data, uživatel s nimi pracuje off-line v terénu a po návratu je synchronizuje s produkční databází. Druhá skupina aplikací žádnou mobilní databázi nemá a připojuje se k centrální pomocí některého z podporovaných protokolů (např. GPRS, WAP, Wifi). Výhody obou způsobů a odstranění nevýhod pak řeší třetí skupina, která je spojením předchozích. Na PDA (kapesní počítač) si nahrajete potřebná data a při práci v terénu můžete výsledky své práce odesílat rovnou do firmy nebo naopak získávat aktuální výsledky svých kolegů.[10]

V dnešní době velkého rozvoje a zvyšování oblíbenosti Javy mezi programátory, ale hlavně přenositelnosti dané aplikace se nabízí naprogramovat tuto aplikaci právě v tomto jazyce. Cílem mé bakalářské práce bylo zjistit jaké možnosti přináší druhá nejmenší edice společnosti Sun Java 2 Micro Edition. Především možnosti profilu CDC, který je určen zejména pro zařízení typu PDA. V konečné fázi pak vytvořit aplikaci, která se připojí přes rozhraní JDBC k databázovému serveru.

2. Základní koncepce Javy

2.1. Oblast použití

Java je univerzální programovací jazyk a může tak být použita v podstatě pro libovolný druh aplikace. To ji odlišuje od takzvaných skriptovacích jazyků, které jsou určeny pouze pro konkrétní oblast, jako například *JavaSkript* pro ovládání webových prohlížečů, *Perl* pro webové servery a makrojazyky jako *VBA* pro spolupráci aplikací společnost Microsoft. Přesto existují oblasti, kde je zvláště vhodné použít Javu, stejně jako oblasti, kde se to příliš nedoporučuje.

Jednou ze silných vlastností Javy je vazba na Internet. Javu je možné použít jak na straně webového prohlížeče (klienta), tak na straně webového serveru. Možné je to

jednak díky tomu, že je Java nezávislá na konkrétní platformě, a díky standardním dostupným třídám, které nabízejí komfortní podporu internetových protokolů a technologií. Proto je v Javě realizováno velké množství bankovních aplikací z oblasti označované e-comerence.

Dalšími výhodami jazyka je jeho modulárnost, množství standardních programových knihoven a rozhraní a vynikající vývojová prostředí, která lze někdy dokonce získat zdarma. Díky tomu je Java velmi často využívána v projektech, kde je vyžadována vysoká efektivita vývoje aplikace a pravděpodobnost budoucích změn.

Javu lze spustit prakticky všude. Existují prostředí pro spouštění těchto programů na kapesních počítačích (PDA), stejně jako na sálových počítačích. Z tohoto hlediska se jazyk používá tam, kde není předem jisté na jaké platformě nakonec aplikace poběží tzv. škálovatelnost.

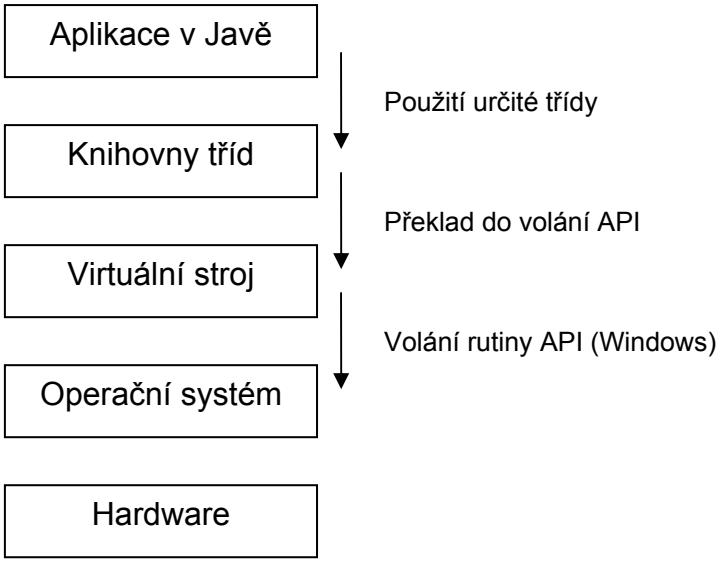
Zvláště oblíbená je Java v distribuovaných objektově orientovaných systémech. Pro řízení veškerých datových toků, rozdělování úloh či zpracování dat je potřeba infrastruktura, která se stará o hladké fungování komunikace (middleware). Java pro tento účel nabízí vlastní mechanismy a technologie jako CORBA (Common Object Request Broken Architekture – standardizovaná infrastruktura pro komunikaci mezi objekty), RMI (Remote Method Invocation - slouží pro vytváření distribuovaných aplikací).

Existuji však i nevýhody Javy a přímé oblasti, kde není vhodné Javu používat. Java nemůže přímo přistupovat k hardwaru. Proto nelze v Javě vytvářet například ovladače zařízení, ale ani adresovat speciální zřízení jako skener či joystick. Nelze vytvářet kritické aplikace vyžadující přístup v reálném čase, které musí garantovat určitou maximální reakční dobu, například 10 milisekund. Zde není možné použít Javu, neboť prostředí pro spouštění programů v Javě je velmi komplexní a v některých situacích jej není možné v konkrétním okamžiku přerušit.

2.2. Java jako platforma

Platforma Java je počítačová platforma zastřešující různé varianty použití programovacího jazyka Java pro vývoj různých typů aplikací. Java obsahuje velmi mnoho již hotových knihoven tříd. Od tříd pro vytváření graficky příjemných uživatelských rozhraní, přes třídy pro přístup k databázím, až po šifrování a kompresi dat. Existuje řada programových rozhraní podobně jako pro operační systémy (např. Windows) ve formě API (Application Programmers Interface, rozhraní pro aplikační

programátory). Ve skutečnosti existuje pro všechny operační systémy jedno a to samé rozhraní a jednotlivá volání jsou za běhu převáděna do skutečné platformy. Toto abstraktně definovaní prostředí spolu s příslušným emulátorem nazýváme virtuální stroj (Java virtual machine). Pro programátora se tak Java jeví jako samostatná platforma, nejen jako programovací jazyk.

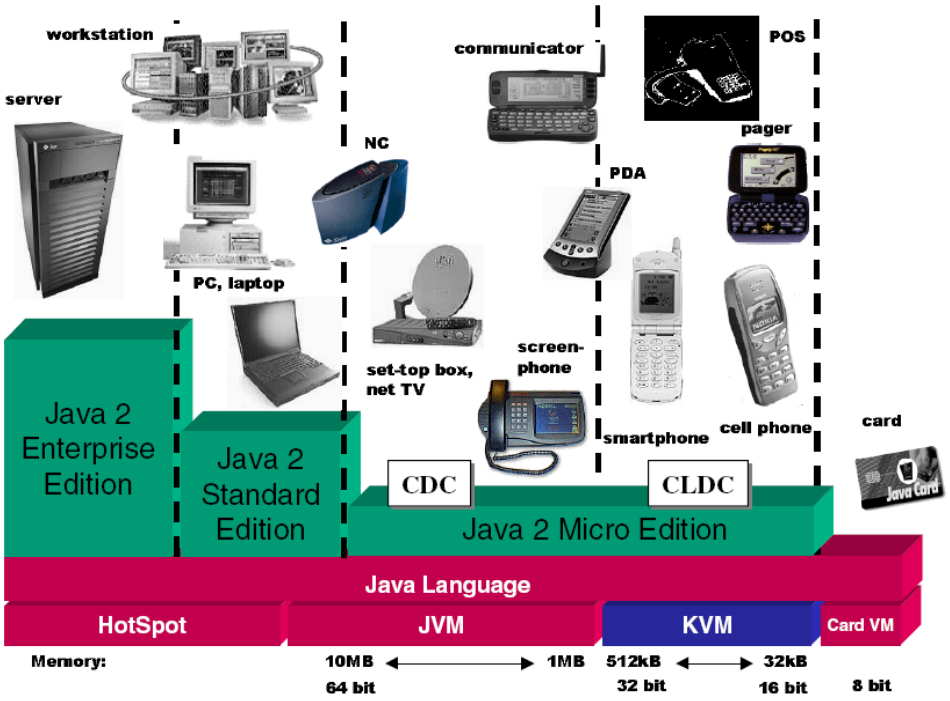


Obr. 2-1 – Překlad

2.3. Základní dělení

Protože je zbytečné nabízet na každé platformě všechny knihovny, například podpora pro komponenty aplikačního serveru je pro PDA zbytečná, rozhodl se Sun (Sun Microsystems – zakladatel jazyka) vymezit několik edic, které se od sebe liší dostupnou funkcionalitou.[2]

- JavaCard – pro aplikace provozované v rámci tzv. „chytrých“ karet (např. platební a kreditní karty atp.)
- Java ME (Micro Edition) – pro aplikace provozované na mobilních zařízeních (mobilní telefony, PDA, atp.)
- Java SE (Standard Edition) – aplikace provozované na stolních počítačích
- Java EE (Enterprise Edition) – aplikace pro podnikové a rozsáhlé informační systémy, aplikační servery



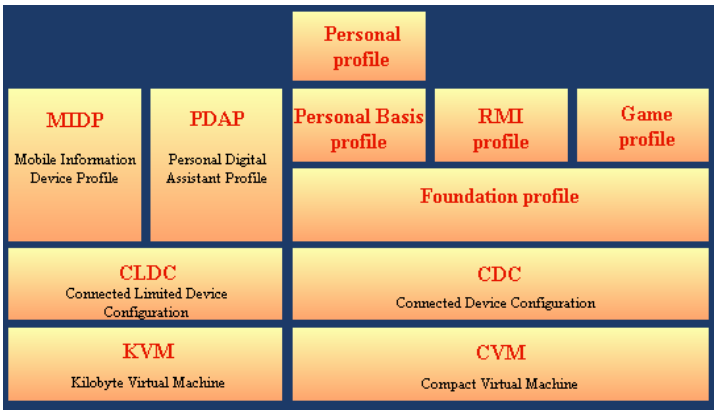
Obr. 2-1 – Edice Javy

Zdroj: http://discolab.rutgers.edu/classes/cs672-fall00/j2me/Appendix1/CLDCappendix-intro.pdf

3. Možnosti J2ME

3.1. Java 2 ME obecně

Java 2 Micro Edition (J2ME nebo Java 2 ME) je druhá nejmenší ze čtyř základních edic Javy. Zaznamenala největší rozvoj s příchodem mobilních aplikací. Cílem této edice bylo sjednotit různé odnože Javy a poskytnout programátorům vývojové prostředí pro tvorbu aplikací určených ke spouštění na malých přenosných zařízeních, jakými jsou například mobilní telefony, pagery, PDA, navigační systémy atd.. Tato zařízení se liší svými vlastnostmi a možnostmi, proto je platforma J2ME rozdělena na různé konfigurace a profily. Konfigurace určuje minimální platformu s podobnými požadavky na operační paměť a výkon procesoru. Určuje základní sadu knihoven, které výrobce daného přístroje může očekávat. Profil tyto vlastnosti dále jen upřesňuje.



Obr.3-1 - Základní členění Java 2 ME

3.2. CLDC (Connected Limited Device Configuration)

Tato konfigurace je zaměřena na malá zařízení. Co se týká hardwaru, specifikuje pouze minimální nároky na paměť, a to 160 kB stálé paměti (např. ROM, při vypnutí zařízení zůstávají data zachována) a 32 kB "nestálé" paměti, která je k dispozici při běhu virtuálního stroje například na zásobník. Z důvodu zmenšení velikosti virtuálního stroje byly některé funkce omezeny nebo odstraněny. Ověřování korektnosti bajtkódu, tzv. verifikaci, kterou obvykle provádí virtuální stroj, bylo rozděleno na dvě fáze. V první fázi (preverifikace), která se provede po překladu zdrojového kódu, je do každé třídy připsán atribut `StackMap`, který pak použije virtuální stroj ke zjednodušení verifikace.

Základní syntaxe jazyka je stejná, jako ve všech ostatních edicích. Knihovny obsahují velmi omezenou podmnožinu standardní edice, a to vybrané a upravené třídy z následujících knihoven:

- `java.lang`

Toto je základní balíček jazyka Java, který obsahuje stěžejní třídy. V programech ho není třeba importovat, je jako jediný balíček importován automaticky.

Nejdůležitější třídy: `Object`, `System`, `Thread`, `Throwable`, `Math`, dále třídy pro datové typy `Number`, `Integer`, `Short`, `Byte`, `Boolean`, `Character`, `Float`, `String`...

Rozhraní: např. `Cloneable`, `Runnable`

Výjimky: `Exception`, `NullPointerException`, `ArrayIndexOutOfBoundsException`, `ClassNotFoundException`, `NumberFormatException`, `RuntimeException`...

- `java.io`

Poskytuje třídy pro práci se vstupy a výstupy pomocí datových proudů a umožňuje práci se soubory. Třídy: `File` (soubor), `InputStream` (vstupní proud), `OutputStream` (výstupní proud), `DataInputStream` (vstupní proud pro čtení standardních typů), `DataOutputStream` (výstupní proud pro čtení standardních typů).

Nejdůležitější výjimky tohoto balíčku jsou: `EOFException`, `IOException`, `FileNotFoundException`.

- `java.util`

Rozmanitý soubor utilit a datových struktur, obsahuje mimo jiné třídy pro reprezentaci data a času, generátor náhodných čísel a základní třídu pro událost. Třídy: `Vector` (rozšiřitelné pole), `Random` (generátor náhodných čísel), `Date` (implementuje časový okamžik), `TimeZone` (práce s časovými zónami), `BitSet` (rozšiřitelné bitové pole), `EventObject` (základní třída, od které všechny události dědí), `Hashtable` (hašovací tabulka)...

A navíc novou knihovnu pro I/O (vstupní/výstupní) operace, s třídami u nichž nebylo možno zachovat podobnost se standardní edicí:

- `javax.microedition.io`

3.3. MIDP (Mobile Information Device Profile)

MIDP je profil, který upřesňuje CLDC konfiguraci pro použití na nejmenších zařízeních, jako jsou obyčejné mobilní telefony. Právě tato platforma se těší největší pozornosti, protože se týká hromadně rozšířených zařízení, tedy právě mobilních telefonů.

Profil MIDP 1.0 definuje oproti CLDC další minimální požadavky (v závorce jsou uvedeny požadavky profilu MIDP verze 2.0):

• na paměť:

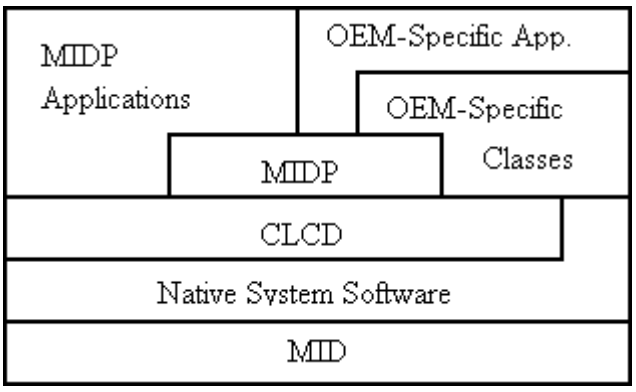
- dalších 128 kB (256 kB) stálé paměti pro komponenty MIDP (knihovny atd.),

- 8 kB stálé paměti pro uchování dat aplikací,
- 32 kB (128 kB) nestálé paměti pro chod Java Virtual Machine a běh programů.

• na displej:
- rozlišení 96x54 bodů (přesně tuto velikost mají například telefony Nokia 3410 nebo 6310i),
- barevná hloubka 1bit,
- poměr šířky a výšky pixelu cca 1:1.

• na vstupní zařízení - alespoň jedno z následujících:
- klávesnice ovládaná jednou rukou,
- klávesnice ovládaná oběma rukama,
- touch screen (dotyková obrazovka).

• na síťové připojení:
- obousměrné, bezdrátové, pokud možno stále k dispozici,
- předpokládá se nízká přenosová rychlost.



Obr. 3-2 – Architektura MIDP

MID - reprezentuje hardware
Native System Software - Operační systém
OEM - specific Classes - definované nad CLCD a MIDP výrobcem zařízení. Jsou určené pro využití speciálních vlastností zařízení. Při použití těchto tříd dojde k omezení přenositelnosti aplikace.

Ke knihovnám specifikovaným v CLDC přidává další:

- `javax.microedition.rms` – správa trvalých dat
- `javax.microedition.midlet` – obsahuje třídu `MIDlet`, která je základní třídou MIDP profilu.
- `javax.microedition.io` – k CLDC přidává třídu `HttpConnection`
- `javax.microedition.lcdui` – třídy pro tvorbu uživatelského rozhraní

Ve verzi MIDP-2.0, jejíž specifikace je již k dispozici, přibude další důležitá funkčnost, jako např. připojení pomocí soketů nebo ovládání zvuku.

Skoro všechny mobilní telefony spadají do kategorie MIDP. Výjimkou jsou některé komunikátory se silnějším procesorem a větší pamětí (např. Nokia 9210). Aplikacím pro tuto kategorii se říká „midlet“ podle základní třídy profilu MIDP.

3.4. PDAP

Profil rozšiřující CLDC konfiguraci, který je, jak už jeho název napovídá, určen pro PDA zařízení s větším displejem. Zatím je teprve ve vývoji.

3.5. CDC (Connected Device Configuration)

CDC je konfigurace cílená na zařízení s 32bitovým procesorem a alespoň 512 kB ROM a 256 kB RAM. Virtuální stroj musí zvládat prakticky stejnou funkčnost jako ve standardní edici.

Knihovny CDC konfigurace tvoří nadmnožinu knihoven CLDC konfigurace. Navíc obsahují knihovny

- `java.net`

Obsahuje třídy umožňující komunikaci po Internetu (stažení dokumentu apod.). Třídy: `URL`, `Socket`, `InetAddress`, `URLConnection`, `HttpURLConnection`, `URLEncoder`...
Výjimky: `ConnectException`, `MalformedURLException`, `ProtocolException`, `SocketException`...

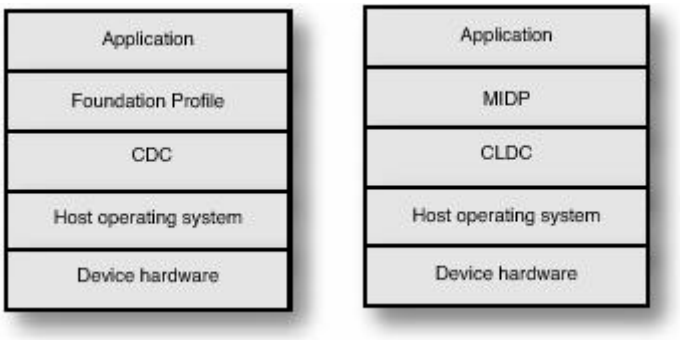
- `java.text`

Umožňuje práci s textem (formátování, parsing - syntaktická analýza atd.)

- `java.security`

Obsahuje obecné třídy a rozhraní pro manipulaci s veřejnými a privátními klíči, digitálními podpisy, certifikáty atd.

Většina následujících specifikací je teprve ve vývoji, proto je následující popis profilů poněkud zkratkovitý.



Obr. 3-3 – Srovnání platforem CDC a CLDC

3.6. Profily

Foundation profil

Přidává většinu základních tříd, které CDC chybí oproti standardní edici. Neobsahuje žádné uživatelské rozhraní a také neobsahuje knihovny `java.beans`, `java.rmi` ani `java.sql` Tvoří základ pro další rozšiřující profily. Vyžaduje 1 MB ROM a 512 kB RAM.

Personal Basis profil

Přidává základní uživatelské rozhraní, které je omezené na použití pouze jednoho okna.

Personal profil

Sem se přesunuje edice Personal java, která je již rozšířená na spoustě zařízení, jako jsou PDA nebo komunikátory (Nokia 9210). Je skoro stejná, jako stará verze Javy před vznikem grafické knihovny `javax.swing`, na kreslení uživatelského rozhraní se používá knihovna `java.awt`. Vyžaduje 2,5MB ROM a 1MB RAM.

RMI profile

Přidává k foundation profilu vzdálené volání metod kompatibilní s rozhraním standardní edice (knihovna `java.rmi`)

Game profil

Ke konečné verzi specifikace má ještě hodně daleko. Měl by být určen speciálně k vývoji her.

3.7. Midlety a jejich omezení

Při běhu se množství paměti, která je k dispozici, pohybuje kolem několika set kilobajtů. Velikost aplikace je na různých zařízeních omezena různě. Pro telefon Nokia 6310 se musí vejít do 30 kB, pro telefon Nokia 7210 do 64 kB, do Nokia 7650 se vejde i 1MB aplikace. Tato čísla jsou tak malá, že je nutné při vývoji věnovat zvláštní pozornost úspornému psaní a neplýtvat pamětí.

J2ME je relativně mladá technologie, což s sebou nese spoustu nedostatků. Přestože vznikla proto, aby sjednotila specifikaci jazyka, je velmi obtížné napsat midlet, který by beze změn běžel na různých telefonech. Pro každý telefon implementuje jeho výrobce J2ME podle téže specifikace. Při pokusu o ověření přenositelnosti aplikace, ale autor zjistí, že se implementace i přesto v některých detailech liší.

Dalším problémem je, že specifikace se některými věcmi nezabývá vůbec – např. ovládáním zvuku, přístupem k SMS, MMS. Tuto funkčnost dodávají výrobci telefonů navíc. Jakmile však vývojář použije rozšiřující třídy, stává se aplikace nepřenositelnou. Dobrou zprávou je, že specifikace MIDP 2.0 už nejzásadnější nedostatky první verze specifikace odstranila a již dnes je zcela běžně podporován mobilními telefony.[7]

4. JDBC

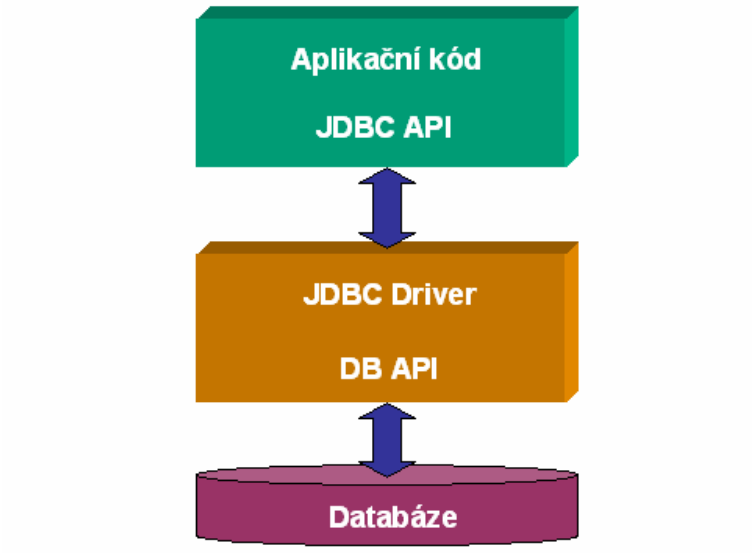
4.1. Úvod do JDBC

JDBC (Java Database Connectivity) je nízkoúrovňové Java API pro spouštění SQL (Structured Query Language) příkazů. Skládá se z množiny tříd a rozhraní napsaných v programovacím jazyku Java. Valná většina rozhraní je uložena v balíku `java.sql`. S využitím tohoto API je např. možné v informačním systému vytvořit jeden program, který bude přistupovat do všech používaných databází např. Oracle, Informix, Sybase. Není tedy nutné pro každý typ databáze vytvářet zvláštní program. Jelikož je JDBC API

rozšířením Java API, je možné např. vytvářet applety, které budou zobrazovat aktuální informace uložené v databázích.

JDBC je záměrně vytvořeno jako "call-level" SQL rozhraní. To znamená, že je zaměřeno na přímé volání SQL příkazů a návrat jejich výsledků. Toto API by mělo jednak sloužit jako základ pro vytváření tzv. higher-level API a současně by mělo najít uplatnění v jeho přímém využití při programování aplikací.

Zjednodušené schéma na obrázku č.4-1 naznačuje základní princip JDBC rozhraní. Logika JDBC je ale složitější, v závislosti na vlastnostech JDBC ovladače.



Obr. 4-1 – Architektura rozhraní JDBC

4.2. Srovnání JDBC a ODBC

V současné době je ODBC pravděpodobně nejrozšířenější programové rozhraní pro přístup do relačních databází, které nabízí přístup do většiny databází. Pomocí Javy je sice možné toto rozhraní používat přímo, ale mnohem výhodnější řešení je používat ODBC pomocí JDBC-ODBC mostu. Přesto, že je tedy možné využívat v Java aplikacích ODBC, je mnoho důvodů proč používat právě JDBC. Hlavní důvody jsou tyto:

- přímý přístup k ODBC z Java aplikací není vhodný, protože je používáno C rozhraní. Volání nativního kódu má mnoho stinných stránek zejména v bezpečnosti, automatické přenositelnosti aplikace, robustnosti a implementaci

- prostý překlad ODBC C API do JAVA API není žádoucí. Java např. nemá pointery a právě ODBC je používá ve velké míře, včetně nechvalně známého pointeru (void *).
- osvojení ODBC je náročné. ODBC vzájemně kombinuje jednoduché a pokročilé vlastnosti a vyžaduje poměrně složitá nastavení pro jednoduché dotazy.
- pokud je používáno ODBC rozhraní, musí být na každém počítači ručně nainstalován ODBC manažer a ovladač. Jestliže je však JDBC napsáno pouze v Javě, je kód ze sítě instalován automaticky, přenositelný a bezpečný na všech Java platformách.

4.3. Dvou-úrovňový a tří-úrovňový model

JDBC API podporuje pro přístup do databází dvou-úrovňový a tří-úrovňový model. Při využití dvou-úrovňového modelu komunikují Java applety a aplikace s databází přímo. Při tomto přístupu je nutné mít pro specifický databázový systém daný JDBC ovladač. SQL příkazy jsou odeslány do databáze a výsledky příkazů jsou odeslány nazpět. Databáze může být umístěna na jakémkoliv počítači, na který má uživatel přístup prostřednictvím sítě (např. Internetu nebo intranetu). V této konfiguraci client/server tvoří klienta hostitelský počítač a server tvoří počítač, na kterém je umístěna databáze.

V tří-úrovňovém modelu jsou příkazy odeslány na střední vrstvu, která odešle SQL příkazy do databáze. Databáze zpracuje SQL příkazy a odešle výsledek zpět na střední vrstvu, která je odešle uživateli. Mezi výhody tohoto modelu patří např. možnost kontroly přístupu k databázím, možnost překladu vyššího API na nízko-úrovňová volání aj.

4.4. SQL

Databázové systémy podporují různé typy SQL syntaxe a sémantiky, které nejsou vzájemně shodné v pokročilých funkcích. JDBC API k tomuto problému přistupuje takto:

- JDBC dovoluje poslat na DBMS (Database Management System) ovladač jakýkoliv řetězec dotazu. Takto může aplikace využít funkčnost, kterou požaduje, s nebezpečím, že může vyvolat chybu na některých DBMS. Ve skutečnosti mohou aplikace používat dotazy, které nejsou SQL, nebo mohou být speciálním deriváty SQL navrženými pro speciální DBMS.
- JDBC poskytuje ODBC styl escape klauzulí.

- Pro komplexní aplikace poskytuje JDBC pomocí rozhraní `DatabaseMetaData` informace, popisující dané DBMS. Tak se můžou aplikace přizpůsobit požadavkům a schopnostem určitého DBMS.

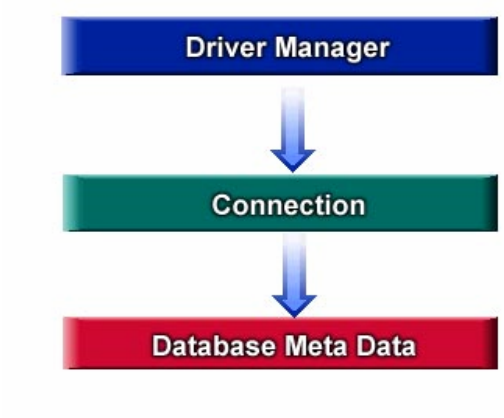
Protože JDBC API tvoří základ pro vyšší databázové nástroje a API, musí JDBC ovladače zajistit určitou standardní úroveň funkčnosti. Tento problém je řešen pomocí označení "JDBC COMPLIANT" – ochranné známky poskytující společností Sun. Takto může být označen pouze ovladač podporující minimálně specifikaci standardu ANSI SQL-2 Entry Level.

4.5. Přehled hlavních rozhraní JDBC

Rozhraní tvoří dvě hlavní skupiny: JDBC API, které je určeno pro aplikační programátory, a nízko-úrovňové ovladače JDBC API.

4.6. JDBC API

JDBC API je specifická skupina abstraktních Java rozhraní, která dovoluje aplikačním programátorům otevřít spojení s databází, spustit SQL příkazy a zpracovat výsledky.



Obr. 4-2 – Průběh jednotlivých rozhraní

Nejdůležitější rozhraní jsou:

- `java.sql.DriverManager`, které spravuje zavádění ovladačů a poskytuje podporu pro vytváření nových databázových spojení
- `java.sql.Connection`, které reprezentuje spojení s konkrétní databází
- `java.sql.Statement`, které představuje kontejner pro spouštění SQL příkazů na daném spojení.
- `java.sql.ResultSet`, které kontroluje přístup k řádkům vráceného výsledku.

Rozhraní `java.sql.Statement` obsahuje dva důležité podtypy: `java.sql.PreparedStatement` pro spouštění předkompilovaných SQL příkazů a `java.sql.CallableStatement` pro spouštění a volání databázových procedur.

4.7. Ovladač JDBC

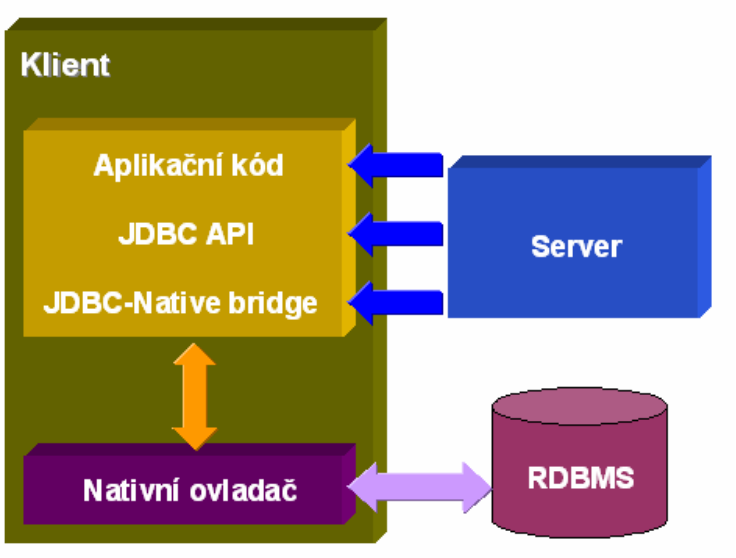
Hlavní úlohou databázových ovladačů je poskytnutí implementace abstraktním třídám JDBC API. Přesněji, každý ovladač musí poskytovat implementaci rozhraní `java.sql.PreparedStatment`, `java.sql.CallableStatement`, `java.sql.Statement`, `java.sql.Connection` a `java.sql.ResultSet`. Dále každý ovladač potřebuje poskytnout třídu, která implementuje rozhraní `java.sql.Driver`. Tato třída je důležitá pro určení ovladače ke konkrétní databázové URL.

4.8. Rozdělení ovladačů JDBC

V současné době je možné ovladače rozdělit do čtyř kategorií:

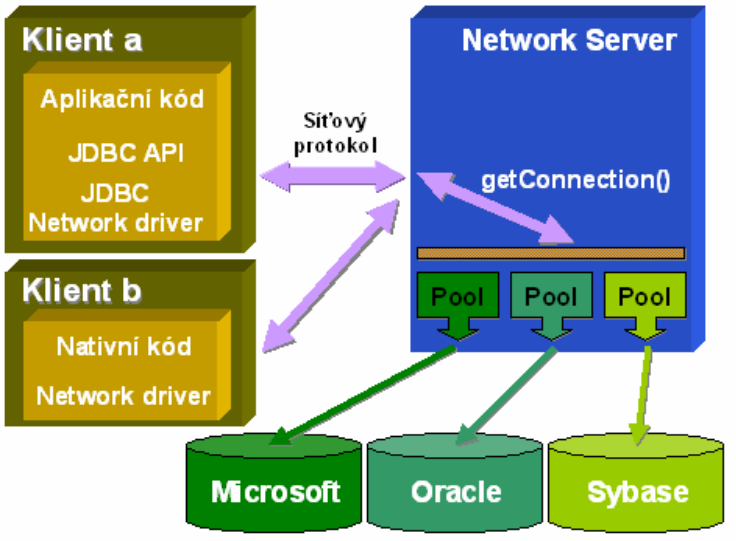
1. JDBC-ODBC bridge plus ODBC driver: Tento typ ovladače využívá lokální ODBC ovladač a přistupuje k němu přes „JDBC-ODBC bridge“. Taková aplikace vyžaduje nainstalování a nastavení lokálního ODBC ovladače pro danou databázi společně s aplikací v Javě, která tento ODBC ovladač využívá. Protože musí být kód ODBC zaveden na každém klientském počítači, který používá tento ovladač, je toto řešení vhodné zejména pro uzavřené sítě, v kterých není instalace na straně klienta hlavním problémem, nebo pro aplikační servery v tří-úrovňovém modelu.

2. Native-API partly-Java driver: Tento druh ovladačů konvertuje JDBC volání na volání klientského API pro Oracle, Sybase, Informix, DB2 a ostatní DBMS. Toto řešení též vyžaduje, aby byl na straně klienta nainstalován určitý binární kód. Dalo by se říci, že „JDBC-ODBC bridge“ je podmnožinou tohoto typu ovladače, s tím, že je čistě vázán jen na ODBC. Pochopitelně se s tímto typem ovladačů pojí stejně výhod a nevýhod jako v případě JDBC-ODBC. Kromě toho ale mohou nastat ještě větší komplikace při administraci a při nasazení.



Obr. 4-3 – Přemostění pomocí překladu na nativní ovladač

3. JDBC-Net pure Java driver: Tento typ již nepoužívá žádný nativní kód pro ovladač, ale je založen čistě na Javě a JDBC, které konvertuje svoji komunikaci do síťového protokolu, který se spojuje s centrálním serverem (Network Server), který poskytuje připojení k databázi. Tento server konvertuje síťový protokol, kterým komunikuje s klienty, do databázově specifického protokolu, jemuž již databáze rozumí. Takový model je vysoce efektivní, a to jak s ohledem na možnost „poolingu“ připojení a tím i zrychlení dotazování a práce s databází, tak i možnost připojení k sadě heterogenních databázových systémů.



Obr. 4-4 – Třetí typ ovladače JDBC

Architektura tohoto typu se obvykle používá v případě rozsáhlých systémů, kde z historických či „politických“ důvodů mohou být rozdílné databázové produkty a network server v podstatě nabízí unifikované rozhraní. Ten se pak pro aplikační programátory tváří jako jedna databáze a je nutné jej pouze jednou nastavit administrátorem takového serveru a následně provozovat.

Kromě toho, k network serveru se mohou připojovat i jiní než Java klienti, díky tomu, že síťový protokol je platformově nezávislý. Tato architektura tedy poskytuje nejen lepší možnosti k optimalizaci výkonu („pooling“), spojení heterogenních databází, ale i provázání heterogenních klientských platforem.

4. Native-protocol pure Java driver: Ovladač typu 4 je napsán celý čistě v Javě s plnou podporou pro optimalizaci vzhledem k dané databázi. Výhodou tohoto ovladače je, že klient nemusí být jakkoli konfigurován a nejsou nutné žádné lokální klientské instalace ovladačů.

Předpokládá se, že pro přístup k databázím budou především preferovány JDBC ovladače typu 3 a 4. Požití ovladačů 1 a 2 je prozatímní řešení v situacích, v kterých ještě není k dispozici ryzí Java ovladač. Dá se říct, že typ 4 je ideální, třebaže existuje několik situací, v kterých je výhodnější typ 3.

4.9. Získání ovladačů JDBC

V současné době existuje spousta ovladačů kategorie 1: ODBC ovladačů, které mohou být použity s mostem od JavaSoftu. Poměrně dost je také ovladačů kategorie 2, které jsou založeny na nativním API pro DBMS. Ovladačů kategorie 3 a 4 je pouze několik, ale velmi rychle se jejich počet rozrůstá. Více informací lze získat na stránkách JBDC na adrese http://www.javasoft.com/products/jdbc. [8] [9]

4.10. Ukázková část programu v Javě

```
String url = "jdbc:mySubprotocol:myDataSource";
String query = "select COF_NAME, PRICE from COFFEES";

try {
Class.forName("myDriver.ClassName");
} catch(Exception ex) {
setError("Can't find Database driver class: " + ex);
return;
}
try {
Vector results = new Vector();
```

```
Connection con = DriverManager.getConnection(url,myLogin",
"myPassword");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(query);
while (rs.next()) {
String s = rs.getString("COF_NAME");
float f = rs.getFloat("PRICE");
String text = s + " " + f;
results.addElement(text);
}

stmt.close();
con.close();
setResults(results);

} catch(SQLException ex) {
setError("SQLException: " + ex);
}
```

5. PDA, operační systém

5.1. PDA

Pod pojmem PDA (Personal Digital Assistant) je souhrnně označována třída „kapesních“ počítačů. Tyto počítače se vyznačují dotykovým displejem a malými rozměry. Tato zařízení většinou nemají klasickou klávesnici (i když k některým zařízením ji lze pořídit) a k ovládání využívají právě dotykový displej. Nedílnou součástí těchto zařízení jsou systémy pro rozpoznávání písma. V dnešní době jsou na trhu s těmito zařízeními dvě dominantní platformy: Palm a PocketPC.

5.2. Palm

Firma Palm vyrábí nejen vlastní zařízení, ale také dodává pro tato zařízení programy, hlavně operační systém označený obchodním názvem PalmOS. Počátek jejich činnosti je podzim roku 1996, kdy firma Palm představila první verzi operačního systému a první zařízení, které bychom dle dnešních kritérií označili jako PDA. Kapesní počítače Palm obsahují procesory firmy Motorola. Aktuálně tyto čipy pracují na frekvencích desítek až stovek MHz. Interní paměť se pohybuje od 8MB. Nejnovější modely obsahují sloty pro přídavné paměťové karty a konektory pro připojení hardwarového rozšíření (např. modul GPS). Standardně je také zajištěna konektivita a možnost synchronizace s PC.

5.3. Pocket PC

Rok po představení prvního přístroje firmy Palm, představil Microsoft nový operační systém Windows CE. Tento produkt vychází z tradičního operačního systému

této firmy Windows. Hlavními požadavky při vývoji systému byly menší paměťové nároky a grafické rozhraní přizpůsobené pro zobrazovací zařízení s poměrně malým rozlišením. Zařízení určená pro provoz Windows CE verze 3.0 a vyšší se nazývají Pocket PC. Zařízení jsou osazována procesory pracujícími na frekvencích řádově ve stovkách MHz. Interní paměť se pohybuje od 32 do 128 megabajtů. Displeje jsou barevné s rozlišením až 640x480 bodů. I tato třída zařízení většinou podporuje přídavné paměťové karty a další hardwarová rozšíření, stejně tak nabízí možnost připojení a synchronizace s osobním počítačem.

Generace operačních systémů pro Pocket PC

Windows CE 2.01 je nejstarší verzí systému pro PPC.

Windows CE Professional (2.11) je poslední systém pro Palm-size PC zařízení. Přinesl některé drobné změny v aplikacích, hlavně však nabídl podporu pro nový hardware a barevné displeje.

Windows 3.0 for Pocket PC je značně vylepšený systém pro zařízení bez klávesnice, jako první nabízí podporu i pro Word a Excel, proti předchozím verzím má značně rozdílné grafické provedení.

Windows 3.0 for Pocket PC 2002 není nová verze operačního systému, je to pouze částečné rozšíření původního systému. Nové grafické a uživatelské rozhraní, každá aplikace je vylepšená. Některá Pocket PC je možné upgradovat, případně je možné zakoupit zcela nové zařízení.

Windows Mobile 2003 (Windows CE 4.2 for Pocket PC) je postaven na zcela novém základě, i když na první pohled vypadá jako verze předchozí. V systému je však plno nových věcí, optimalizoval se kód pro novou generaci procesorů, drobné změny se týkají takřka všech aplikací, některé aplikace jsou zcela nové.

Windows Mobile 2003 Second edition (Windows CE 4.2 for Pocket PC) jen lehce rozvíjí základní verzi systému. Nově je přidána podpora pro jiná rozlišení displeje než klasické 240x320 bodů, systém začal nativně podporovat okamžitou změnu orientace obrazovky. Vylepšení se týká také některých aplikací, například Pocket IE.

Windows Mobile 5 (Windows CE 5.0 for Pocket PC) přináší revoluční změnu v práci s pamětí, podporu pro HW klávesnic, vylepšený Mobile Office, 3G komunikaci a další novinky.

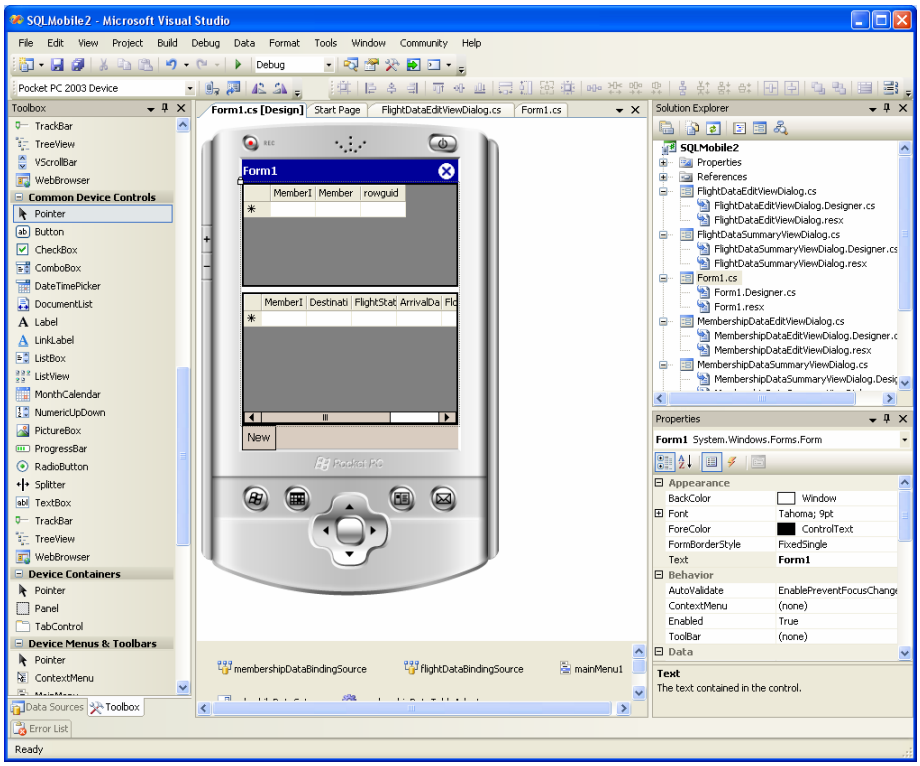
6. Vývoj aplikace pro PDA

6.1. Vývoj aplikací pro PDA obecně

PDA má mnohem větší možnosti, než většina ostatních mobilních zařízení jako třeba mobilní telefony. Jde zejména o velikost paměti, rychlost zpracování instrukcí a v neposlední řadě také velikost zobrazovací plochy. První, co je třeba při tvorbě aplikace provést, je zvolit prostředí (platformu), na níž bude celá aplikace stát. Převážná většina softwaru vyvíjeného pro Pocket PC je vytvářena na platformě .NET. Vzhledem k přenositelnosti a dalším přednostem Javy se budeme zabývat vývoji právě na této platformě.

6.2. .NET

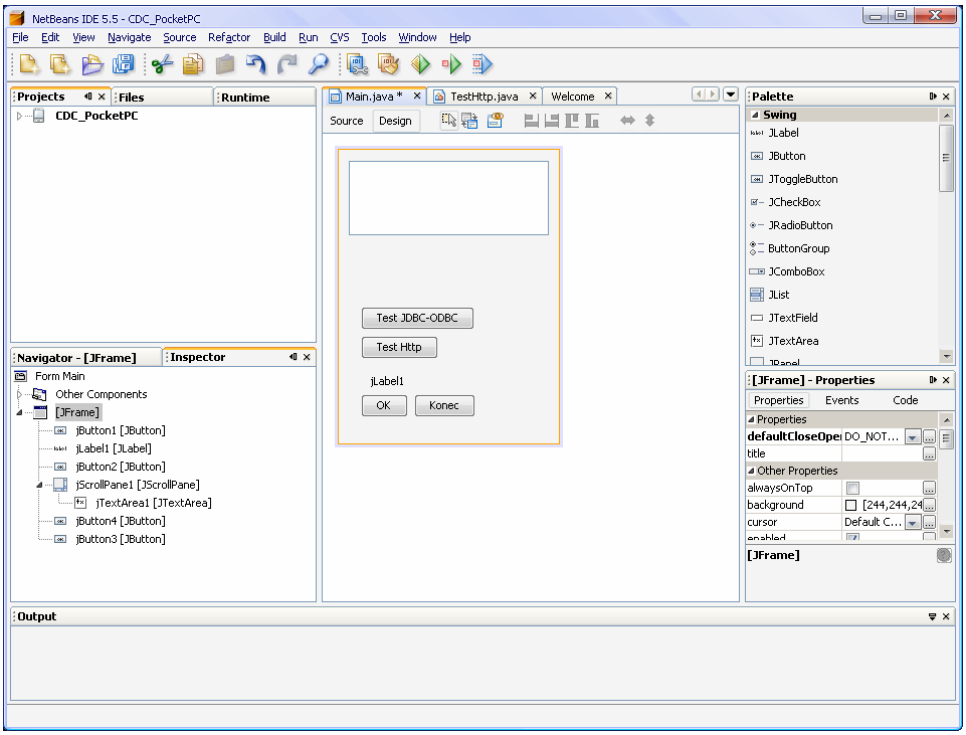
Při návrhu a vývoji aplikace v prostředí .NET je k dispozici nástroj *MS Visual Studio*. Zde lze celkem rychle a přehledně tvořit aplikace všeho druhu. Jedná se o kompletní prostředí s editorem zdrojových textů, GUI editorem, debuggerem a kompilátorem. Při tvorbě aplikace je třeba vybrat nejprve typ aplikace (resp. typ celého projektu). Pokud to má být aplikace pro PDA stačí vybrat typ projektu *Smart Device Application*. Po výběru typu zařízení je třeba ještě zvolit typ aplikace. Zda má být pro Pocket PC nebo Windows CE, což jsou asi dva nejpodporovanější operační systémy a o jaký druh aplikace se jedná (Windows application, class library, non-grafical application). Pak už nezbývá než aplikaci naimplementovat. K ulehčení práce s formuláři je k dispozici také form editor, kde lze „naklikat“ kompletní grafické rozhraní aplikace. Poklepáním na některý z prvků formu se zobrazí textový editor, kde je možně dopsat kód k danému prvku, který se provede, když nastane příslušná událost, jako kliknutí na tlačítko a podobně. Výslednou aplikaci je možné spustit v emulátoru nebo přímo nainstalovat a spustit v PDA. Emulátor kompletně napodobuje operační systém Pocket PC (resp. Windows CE) a je v něm možné provádět stejné operace jako na PDA.



Obr. 6-1 – Microsoft Visual Studio

6.3. Java

S podporou Javy je to u Pocket PC mnohem složitější než u .NETu. Standardně v podstatě žádná podpora není. Pokud na zařízení má být provozován Java kód, je nutné nejprve obstarat virtuální stroj a ten do zařízení nainstalovat. Na trhu je řada JVM pro Pocket PC, některé komerční, některé zdarma. Obvykle však platí, že produkty zdarma podporují starší verze Javy. Nicméně je možné na Internetu najít vhodný virtuální stroj a do zařízení jej nainstalovat. Co se týče vývojového prostředí, je možné jich najít mnoho a část z nich je i pro nekomerční použití zdarma. K nim patří NetBeans, JBuilder, Elipse a další. Některé z nich nabízejí, podobně jako MS Visual Studio, návrh vzhledu ve form editoru. V případě vývojového prostředí NetBeans, v kterém bude aplikace vyvíjena, lze doinstalovat jednotlivé profily a konfigurace tzv. Add-ons pro kterou chceme programovat konkrétní program. Tento balíček zajistí programátorovi, aby byly použité stejné knihovny specifické pro danou konfiguraci (CDC, CLDC atd.) jako na virtuálním stroji PDA. Co se týče emulátorů, s tím to je trošku horší. V době vytváření dané aplikace se nám nepodařilo najít funkční PocketPC emulátor, který by podporoval Javu. Jak již bylo zmíněno JVM je nutno do PDA instalovat dodatečně. Vytvořenou aplikaci jsem tedy testoval přímo na zařízení. Později na emulátoru od společnosti NSIcom zvaném CreMe. [11]



Obr. 6-2 - IDE Netbeans 5.5 s CDC platformou

7. Vývojové prostředí

7.1. NetBeans IDE

Vývojové prostředí NetBeans IDE je nástroj, pomocí kterého programátoři mohou psát, překládat, ladit a šířit programy. Vývojové prostředí je vytvářeno v jazyce Java - ale může podporovat jakýkoliv programovací jazyk (např. C++). Kromě toho také existuje velké množství modulů, které toto vývojové prostředí rozšiřují. Vývojové prostředí NetBeans je bezplatně šířený produkt, který je možné používat bez jakýchkoliv omezení. V průběhu psaní této bakalářské práce došlo k vytvoření novější verze NetBeans 5.5 a zároveň i příslušného rozšiřujícího balíčku pro CDC.

7.2. Mobility Pack pro CDC

Tento balíček nám zajistí vytváření aplikací právě pro tento profil. Před vytvořením nového projektu je zapotřebí nainstalovat emulátor (platformu pro kterou chceme programovat), který nám zajistí přeložení kódu a zobrazení grafického uživatelského prostředí. Tento emulátor je nezbytný pro odladění aplikace na PC. Emulátorů pro tento profil v době psaní této práce je několik. To ovšem neplatilo na počátku této práce. Následkem bylo obtížné testování funkčnosti programu a zdržení celkového vývoje.

Nás zajímá především platforma PocketPC resp. Windows CE. Tento emulátor je založen na VM jménem CrEme for Windows CE vytvořenou společností NSIcom.

Celý trik spočívá v tom, že si nainstalujeme JVM CrEme na počítač, kde máme prostředí NetBeans. Přeložený program v projektu CDC se pak automaticky spustí pod touto JVM. Aby aplikace fungovala naprosto korektně je dobré mít nainstalován stejnou JVM i na daném zařízení (PDA). Není to ovšem nutností, protože pak by aplikace ztrácela na přenositelnosti a nevyužívala by přednosti jazyka Java.

7.3. Instalace CDC profilu a emulátoru Windows CE

Z důvodu počátečních problémů s instalací NetBeans starší verze 5.0 a příslušného CDC packu uvedu postup instalace s již novější verzí programu. Problém se týkal především emulátoru pro profil CDC, který byl určen pro multifunkční komunikátory Sony Ericsson M600 a M990 pracující s operačním systémem Symbian podporující J2ME.

K tomu abychom nainstalovali Mobility Pack pro CDC 5.5.1 je zapotřebí mít nainstalovaný soubor základních nástrojů pro vývoj aplikací pro platformu Java nebo-li Java SE Development Kit (JDK) 5.0 nebo 6.0 a vývojové prostředí NetBeans 5.5.1. Nyní nainstalujeme samotný Mobility Pack.

- **Java SE Development Kit (JDK) 5.0 or 6.0** (http://java.sun.com/javase/downloads/index.jsp)
- **NetBeans IDE 5.5.1** (http://www.netbeans.info/downloads/index.php?rs=21)
- **NetBeans Mobility Pack 5.5.1 for CDC** (http://www.netbeans.info/downloads/index.php?rs=21&p=10)

Jak již bylo zmíněno v předchozí části, použijeme emulátor Windows CE od společnosti NSIcom (CrEme for Windows CE). Nejprve nainstalujeme samotnou JVM CrEme.

- **CrEme 4.10 Developer Support** (download - http://nsicom.com/shared/CrEmeDevSup410.exe)

Přidání nové platformy v prostředí NetBeans provedeme pomocí *Main Menu > Tools > Java Platform Manager*. Přidat platformu (*Add Platform*), vybereme *Java Micro Edition CDC Platform Emulator* a klikneme na *Další*. Dále vybereme adresář kde se nachází platforma SDK. Standardní cesta je: *C:\Program Files\NSIcom\CrE-ME V4.10*. Klikneme na *Další*. Nyní by měla být CDC platforma registrována v IDE.

Pro testování můžeme použít přímo emulátor zařízení PocketPC se systémem Windows Mobile 2003. Je nutné doinstalovat:

- **Microsoft Device Emulator 1.0** (download - http://www.microsoft.com/downloads/details.aspx?FamilyId=C62D54A5-183A-4A1E-A7E2-CC500ED1F19A&displaylang=en#Instructions)
- **Virtual Machine Network Driver for Microsoft Device Emulator** (download - http://www.microsoft.com/downloads/details.aspx?familyid=DC8332D6-565F-4A57-BE8C-1D4718D3AF65&displaylang=en)
- **ActiveSync** (download - http://www.microsoft.com/downloads/details.aspx?FamilyID=9e641c34-6f7f-404d-a04b-dc09f8141141&DisplayLang=en)

8. SQL Server

Abychom mohli danou aplikaci testovat, je zapotřebí nainstalovat na příslušný počítač databázový server. Případně vytvořit jednoduchou tabulku – databázi a práva uživatele. Pro testování připojení k databázi byl použit SQL Server společnosti Microsoft, přesným názvem Microsoft SQL Server 2005.

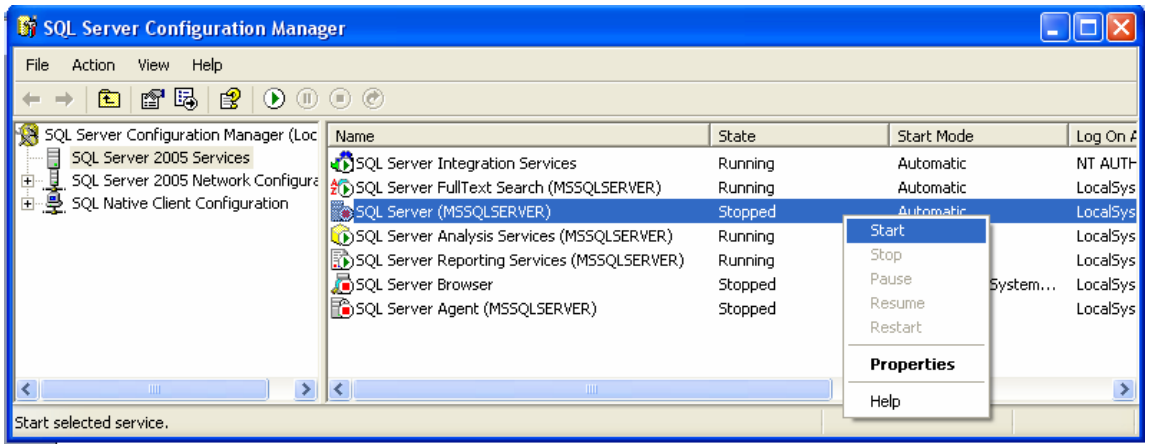
Ke konci práce se ukázalo, že tato volba se stala velmi užitečnou. Tento server jsme totiž rozšířili o SQL Server Mobile Edition a mohli tak otestovat řešení společnosti Microsoft. O tom se dočteme v další části.

8.1. Instalace MS SQL Serveru

Instalace samotného serveru probíhá prakticky bez sebemenších problémů. Instalátor vám dá možnost na výběr zda chcete všechny součásti a nástroje. V našem případě zvolíme plnou instalaci. Je nutné připomenout, že musíme instalovat plnou verzi. Verze nabízené volně ke stažení na internetových stránkách Microsoftu jsou v určité míře omezené a neumožňují nám vytváření nových publikací.

8.2. Potřebná nastavení

Po instalaci nastavíme způsob spouštění serveru. V záložce *Start > Microsoft SQL Server 2005 > Configuration Tools* spustíme *SQL Configuration Manager*. V nastavení můžeme zvolit automatické, vypnuto nebo manuální spouštění. Z hlediska testování mohu doporučit spouštění manuální.



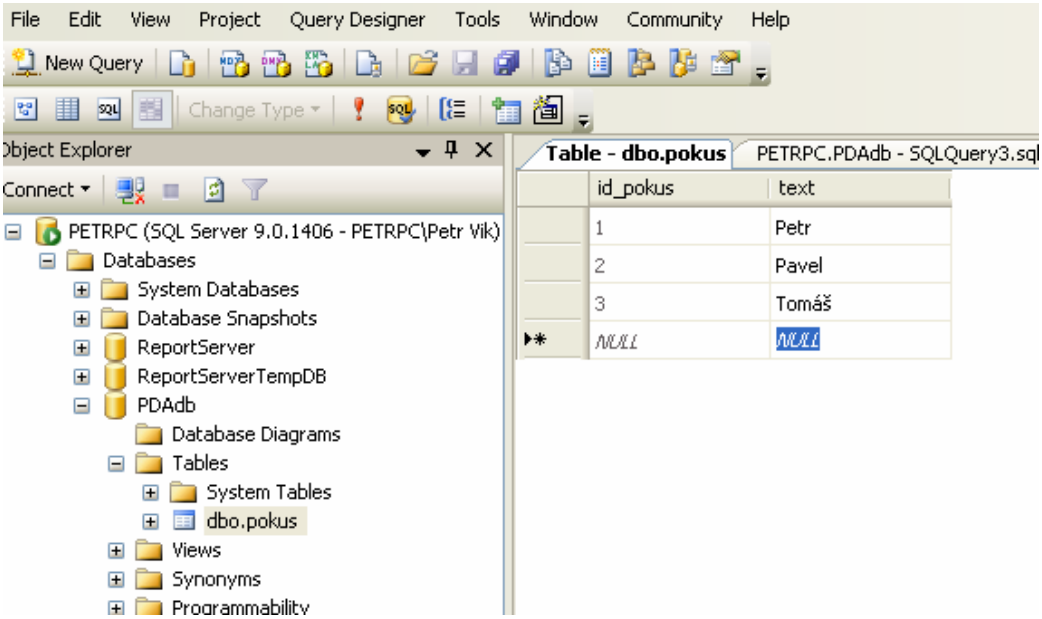
Obr. 8-1 – Manažer konfigurace

8.3. Vytvoření databáze

Spustíme server. Nyní v záložce *Start* > *Microsoft SQL Server 2005* spustíme *SQL Server Management Studio*. V okně připojení k serveru vybereme *Windows Authentication* a klikneme na *Connect*. V *Object Exploreru* v kořenové nabídce klikneme pravým tlačítkem na *Databases* a vybereme *New Database*. Zadáme jméno např. „PDAdb“ a klikneme na *OK*. Pod položkou *Database* se nám vytvořila nová databáze pod daným jménem. Označíme ji a pravým tlačítkem na *Tables* vybereme *New Table*...Zde si můžeme určit jednotlivé proměnné a stanovit délku jednotlivých záznamů. Novou tabulku vytvoříme i pomocí skriptu. V panelu nástrojů klepneme na tlačítko *New Query* . Následující skript nám vytvoří tabulku v databázi *PDAdb* pod jménem *pokus*.

```
USe PDAdb;
create table pokus (
id_pokus int IDENTITY(1,1) NOT NULL,
text text,
CONSTRAINT [PK_pokus] PRIMARY KEY CLUSTERED
(
[id_pokus] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

Nově vytvořenou tabulku otevřeme pomocí *Open Table* a dosadíme testovací data. Dále klepneme na *Execute SQL* (červený vykřičník).



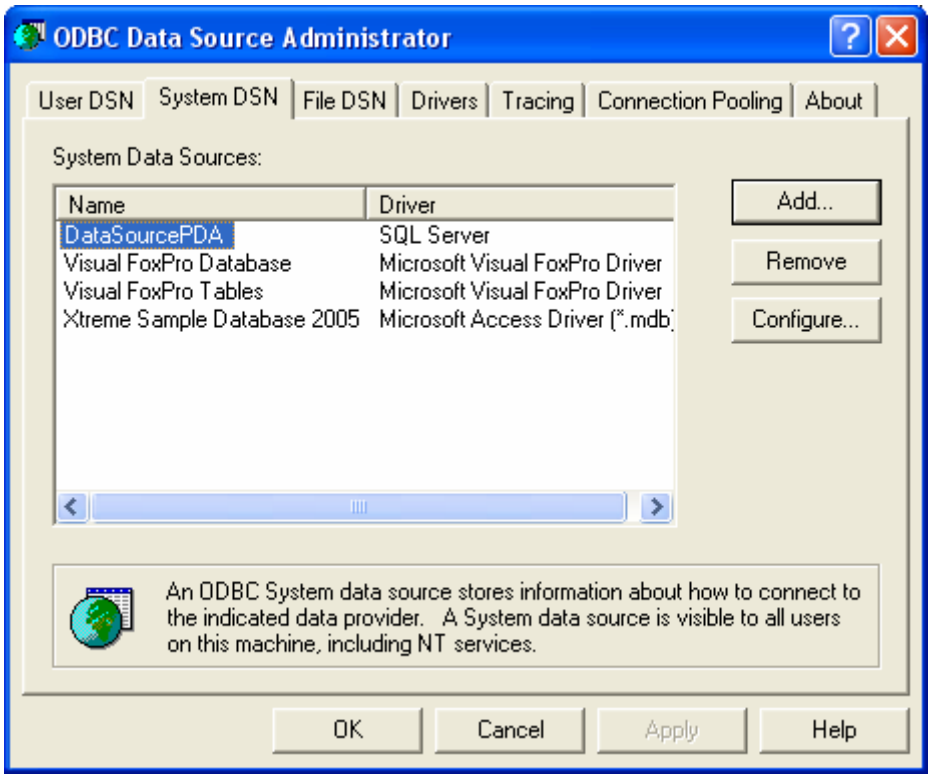
Obr. 8-2 – Vytvoření tabulky, zadaní dat

8.4. Nastavení autentifikace a práv uživatele

Dále je potřeba povolit SQL autentifikaci na serveru. V *Object Explorer* pravým tl. myši na server (úplně nahoře ve stromu) *Properties* v záložce *Security* zvolíme *SQL server and windows autentification mode*. Dále v záložce Security v kořenovém adresáři pravým tl. založíme nového uživatele a nastavíme mu *SQL Server authentication*. Dále na kartě *Server Roles* nastavíme práva *dbcreator*. Tím zajistíme uživateli práva pro vytváření nových databází. Nyní je ještě nutné nastavit uživatele přímo pro danou databázi. V našem případě pro PDAdb. Otevřeme databázi a na položce *Security* vytvoříme nového uživatele se stejným jménem a nastavíme práva *db_datareader*.

8.5. Nastavení ODBC

Posledním krokem bude nastavení ODBC (Open Database Connectivity). Přidání zdroje provedeme pomocí *Ovládací panely* > *Nástroje pro správu* > *Data SourcesODBC*. Na záložce *System DNS* klikneme na přidat, vybereme *SQL Server*. Dále vyplníme jméno a server: (local)\MSSQLSERVER případně jen jméno PC. Zvolíme ověření SQL Serveru pomocí uživatelského jména a hesla, vyplníme heslo a jméno. Dále již necháme přednastavené nastavení.



Obr. 8-3 – Nastavení zdroje dat

9. Program v Javě

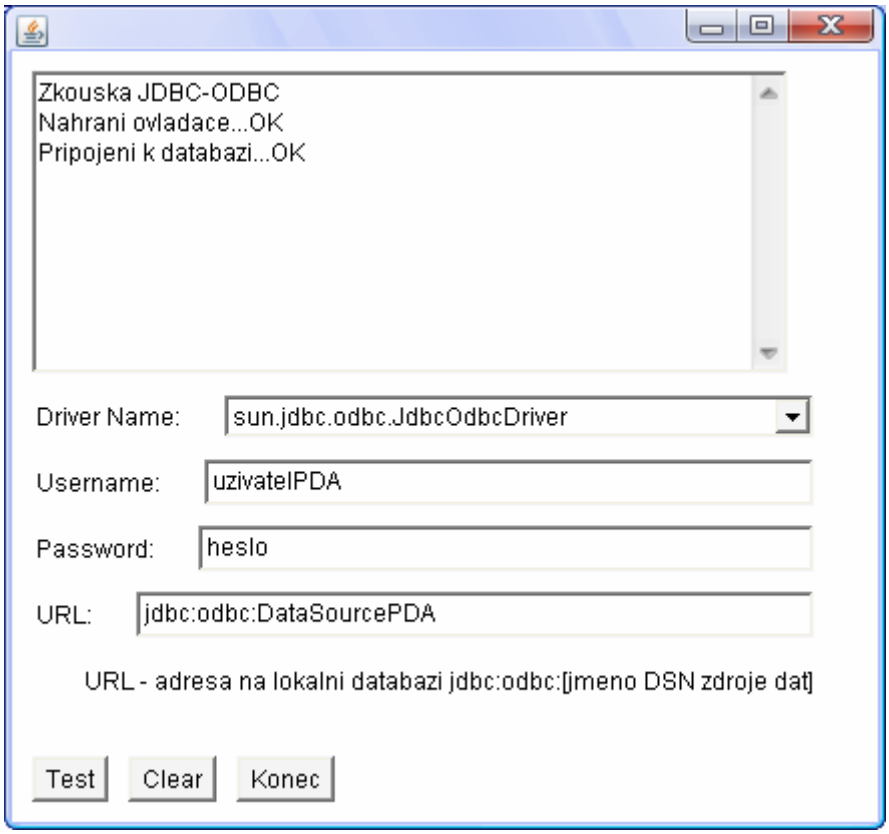
Vzhledem k prvotním problémům souvisejících s nainstalováním CDC profilu a s nalezením odpovídajícího emulátoru co nejblíže připomínající specifikaci PDA je první část věnována programu vytvořeného pro základní edici Java 2SE. Druhá část popisuje pokus o vytvoření programu pro J2ME s CDC profilem využívající emulátor CrEme a problematiku týkající se nefunkčnosti natažení ovladače příslušné databáze přímo z emulátoru a spojení s databází.

9.1. Program v J2SE

Program využívá následující třídu:

`java.sql`

Je vytvořeno jednoduché grafické rozhraní pro ovládání a nastavení aplikace. Veškeré výstupy jsou směrovány do textového pole umístěného v horní části okna.



Obr. 9-1- Výsledná aplikace

Pro připojení k databázi potřebujeme nejprve ověřit, zda uživatel má nainstalován požadovaný ovladač. Pokud ano, můžeme využít statické tovární metody `DriveManager` a získat spojení do databáze voláním metody `getConnection()`. Metoda `getConnection()` existuje ve třech přetížených variacích, ve kterých přijímá parametry potřebné pro vytvoření spojení do databáze. Typicky je nutné předat uživatelské jméno a heslo.

Nastavení parametrů:

```
Jméno ovladače: ("sun.jdbc.odbc.JdbcOdbcDriver");
Uživatelské jméno: ("uzivatelPDA");
Heslo: ("heslo");
Adresa zdroje dat URL: ("jdbc:odbc:DataSourcePDA");
```

Zdrojový kód realizující natažení příslušného ovladače databáze:

```
//nahrani ovladace-----------------------------------------------------
        textArea1.append("Nahrani ovladace...");
        try{
            //ovladac pro JDBC-ODBC most
"sun.jdbc.odbc.JdbcOdbcDriver";
            String driverName = choice1.getSelectedItem();
            Class.forName(driverName);
            textArea1.append("OK\n");
        }
        catch (ClassNotFoundException e){
            textArea1.append("Error\n");
```

```
            textArea1.append("Ovladac pro SQL Server nebyl
nalezen\n");
        }
```

Volání `Class.forName()` s parametrem plně kvalifikovaného názvu třídy ovladače způsobí natažení této třídy do paměti. Navíc přitom dojde v tomto případě databázových ovladačů k jejich registraci do prostředí. Proto považujme volání `Class.forName()` za bezpečný způsob zajištění nahrání potřebného kódu do paměti.

Zdrojový kód realizující připojení k databázi:

```
//pripojeni k databazi----------------------------------------
        textArea1.append("Pripojeni k databazi...");
        Connection con = null;
        try{
            String username = textField2.getText(); // uzivatelPDA
            String password = textField1.getText(); // heslo
            String url2 = textField3.getText();     //
jdbc:odbc:DataSourcePDA
            con = DriverManager.getConnection(url2,username,password);

            con.close();
            con = null;
            textArea1.append("OK\n");
            textArea1.append("Pripojeni bylo navazano");
            spojeni = true;
        }
        catch (SQLException e){
            //System.out.println("Error");
            textArea1.append("Error\n");
            //System.out.println("Nepodarilo se pripojit k
databazi.");
            textArea1.append("Nepodarilo se pripojit k databazi.\n");
            e.printStackTrace();
            spojeni = false;}
```

9.2. Program v J2ME

Při vytváření podobného programu pro J2ME v profilu CDC se objevily následující komplikace:

- Metoda `DriveManager` není běžnou součástí profilu CDC. Není tedy možné volat metodu `getConnection()` a navázat tak spojení. V případě emulátoru CrEme je součástí (nesplňuje přesnou specifikaci CDC).
- Pomocí metody `Class.forName()` nelze nahrát odpovídající ovladač.
- Aplikace se nespojí s databázovým serverem resp. se zdrojem informací přes most JDBC – ODBC.

Jelikož se tímto nepodařilo splnit daný bod zadání, bylo třeba navrhnout alternativní řešení zabývající se podobným problémem. K vypracování tohoto úkolu již nezbývalo mnoho času a tak po dohodě s vedoucím práce došlo na řešení od společnosti Microsoft podporující mobilní databázové aplikace.

10. SQL Server CE

Microsoft SQL Server CE (přesný název aktuální verze je Microsoft SQL Server 2005 Mobile Edition) je databázový stroj, který je určen pro zařízení založená na operačních systémech Microsoft Windows CE 5.0, Microsoft Windows XP Tablet PC Edition, Windows Mobile 2003 Software for Pocket PC a Windows Mobile 5.0.

Výhodou tohoto řešení je velká podpora z hlediska vývoje (přímá podpora v MS Visual Studio a .Net Compact Framework) i provázání s operačním systémem mobilních zařízení na MS platformě. Velikou výhodou oproti konkurenci je možnost pracovat jak s off-line zařízeními při občasné synchronizaci s podnikovou databází, tak zároveň i se zařízeními přímo připojenými, kdy se datové operace provádí místo nad lokální kopii dat přímo nad podnikovou databází. Přístup je z hlediska aplikace transparentní.

Bezpečnost dat při přenosu mezi serverem a klientskými zařízeními je zajištěna prostředky IIS serveru – je tedy možné využít protokol HTTPS s nutností zadání uživatelského jména a hesla a případně i autentizaci uživatele v rámci domény Windows.

Jednou z hlavních nevýhod je omezení pouze na zařízení s operačním systémem z dílny Microsoftu, propojení výhradně s velkou MS SQL databází a pak také fakt, že toto řešení nepodporuje distribuci aplikačních souborů na mobilní zařízení. [12]

V následující časti bude popsán popis nastavení serveru a v konečné fázi vytvoření samotné aplikace. Tento návod je součástí technické dokumentace *Microsoft SQL Server 2005 Mobile Edition*.

10.1. Instalace

Předpokladem pro instalaci SQL Server CE a vytvoření obslužného programu na PDA je mít operační systém Windows XP, aktivní IIS službu (Internet Information Services), Visual Studio 2005 a samozřejmě SQL Server 2005. Na stránkách Microsoftu stáhneme *Microsoft SQL Server 2005 Mobile Edition Device SDK* a nainstalujeme.

10.2. Vytvoření publikace na serveru

Před vytvořením vlastní aplikace musíme nejdříve nakonfigurovat publikaci v SQL Serveru 2005. Nejprve však vytvoříme pomocí následujícího skriptu zkušební databázi, kterou budeme následně synchronizovat.

```
USE master;
GO
DROP Database SQLMobile;
GO
CREATE DATABASE SQLMobile;
GO
USE SQLMobile;
GO
CREATE TABLE MembershipData (MemberID INTEGER IDENTITY (1,1) CONSTRAINT pkMemberID
PRIMARY KEY, MemberName NVarChar (50));
CREATE TABLE FlightData (MemberID INTEGER FOREIGN KEY REFERENCES
MembershipData(MemberID), Destination NVarChar (50), FlightStatus NVarChar(50),
ArrivalDate DATETIME, FlownMiles INTEGER);
INSERT INTO MembershipData (MemberName) VALUES ('Mr Don Hall');
INSERT INTO MembershipData (MemberName) VALUES ('Mr Jon Morris');
INSERT INTO MembershipData (MemberName) VALUES ('Ms TiAnna Jones');
INSERT INTO FlightData (MemberID, Destination, FlightStatus, ArrivalDate, FlownMiles)
VALUES (1, 'Seattle', 'Flight Delayed 1 hour', '8/25/00', '20000');
INSERT INTO FlightData (MemberID, Destination, FlightStatus, ArrivalDate, FlownMiles)
VALUES (2, 'London', 'Flight on time', '9/12/00', '15000');
INSERT INTO FlightData (MemberID, Destination, FlightStatus, ArrivalDate, FlownMiles)
VALUES (3, 'Sydney', 'Flight Gate Closing', '11/5/00', '30000');
INSERT INTO FlightData (MemberID, Destination, FlightStatus, ArrivalDate, FlownMiles)
VALUES (1, 'Tokyo', 'Delayed Fog', '5/25/00', '25000');
INSERT INTO FlightData (MemberID, Destination, FlightStatus, ArrivalDate, FlownMiles)
VALUES (2, 'Minneapolis', 'Flight on time', '5/1/00', '1000');
INSERT INTO FlightData (MemberID, Destination, FlightStatus, ArrivalDate, FlownMiles)
VALUES (3, 'Memphis', 'Flight Gate Closing', '1/5/00', '1000');
```

Tento skript vytvoří databázi jménem *SQLMobile*, která obsahuje dvě tabulky. První *MebershipData* obsahuje seznam jmen pilotů, druhá *FlightData* obsahuje letecká data (ID pilota, cíl, stav, datum, počet mílí).

Dále je nutné vytvořit uživatelský účet snapshot agenta, pod kterým bude vytvořen adresář určený pro sdílení souborů určených k synchronizaci. Ve složce *Start* > *Nastavení* > *Ovládací panely* > *Nástroje pro správu* > *Správa počítače* vytvoříme nového uživatele s následujícími údaji.

Položka	Hodnota
Uživatelské jméno	snapshot_agent
Heslo	p@ssw0rd
Ověření hesla	p@ssw0rd
Při dalším přihlášení musí uživatel změnit heslo	Ne
Heslo je platné stále	Ano

Vytvoříme adresář s názvem „snapshot“ a nastavíme sdílení a oprávnění. Jako název objektu zadáme *computername\snapshot_agent* kde *computername* je název počítače. Oprávnění nastavíme na *změnit* a *čtení.*

Samotné vytvoření publikace se provede v *Object Exploreru v SQL Server Mangement Studiu*. Otevřeme položku *Replication* a pravým tlačítkem myši na *Publication* zvolíme novou publikaci. Jako cestu distributora dat nastavíme cestu k právě vytvořenému adresáři. Tedy *\\computername\snapshot*. Z výběru databází vybereme SQLMobile. Z typů publikací vybereme *Merge publication*. Aktivujeme podporu pro SQL Server Mobile. Zaškrtneme publikaci tabulek a na stránce *Snapshot Agent* ponecháme defaultní nastavení. V dialogovém okně *Agent Security* v nastavení zadáme jméno *computername\snapshot_agent* a heslo *p@ssw0rd*. Tímto máme vytvořenou publikaci. Zabezpečení publikace nastavíme na Windows Authentification.

10.3. Konfigurace IIS a web synchronizace

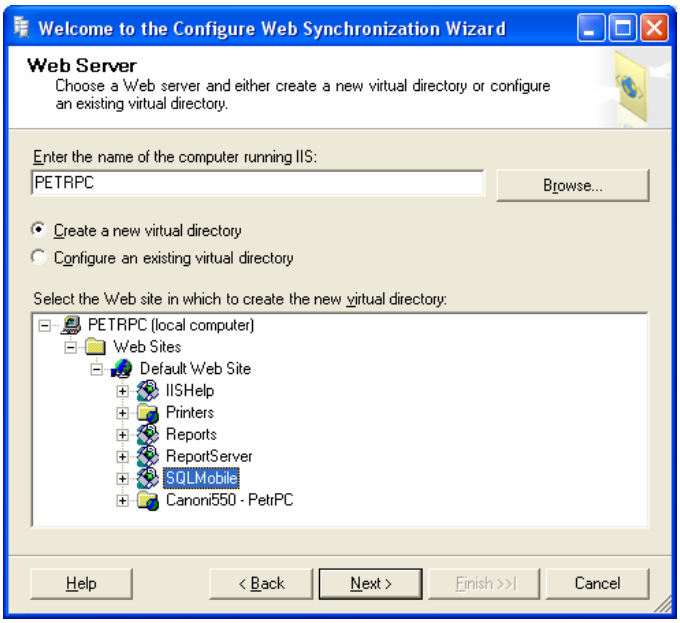
Aby byl SQL Server přístupný pro mobilní klienty, musíme publikaci zpřístupnit. SQL Server Mobile spojuje SQL Server pomocí služby IIS. Vytvoříme a nakonfigurujeme tedy virtuální adresář vytvářející serverového agenta přístupného klientům.

Nejprve je zapotřebí nainstalovat komponenty SQL Server Mobile. Instalační soubor komponent *sqlce30setupen.msi* najdeme v adresáři:

C:\Program Files\Microsoft SQL Server\90\Tools\Binn\VSShell\Common7\IDE.

Konfiguraci publikace pro synchronizaci přes web provedeme spuštěním průvodce, který se nachází ve složce *Start > Programy > SQL Mobile > Configuration Web Synchronization Wizard*.

Druhé jméno (neboli alias) virtuálního adresáře nastavíme na „SQLMobile“ a sdílení informací navedeme na cestu *\\computername\snapshot*.



Obr. 10-1 – Konfigurace webové konfigurace

10.4. Vytvoření nové SQL Server Mobile databáze

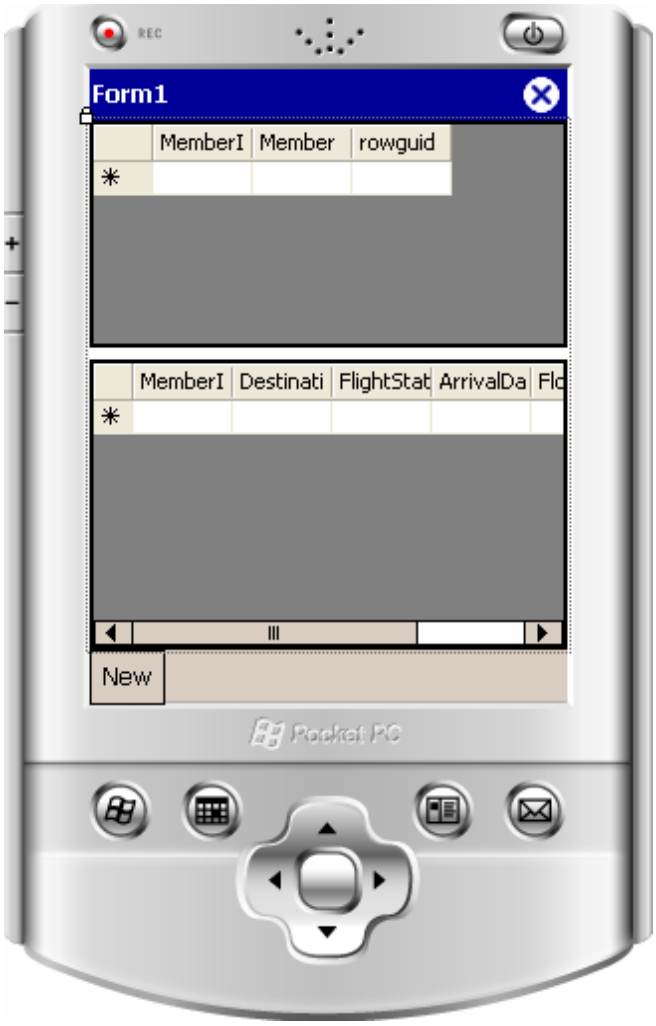
Pro ulehčení práce s vytvářením a manipulaci s daty v mobilní databázi nám SQL Server Managment Studio dovoluje připojit se k SQL Server Mobile a pracovat tak s databází na lokálním počítači. V *Object Exploreru* při připojování zvolíme *SQL Server Mobile*, vybereme *<New Database…>* , jméno souboru *c:\sqlmobile.sdf* a připojíme se k serveru. Zbývá vytvořit nový příspěvek tzv. *subscription*. V nastavení *Web Server Authentification* zvolíme adresu virtuálního adresáře, který byl vytvořen v minulých krocích *http://localhost/SQLMobile.* Po dokončení autentifikace v konečném okně je dobré si zkopírovat příslušný kód, který nám zajišťuje přístup k databázi.

10.5. Vytvoření aplikace

Samotné vytváření konkrétní aplikace je provedeno pomocí vývojového prostředí Visual Studio 2005, o kterém bylo již zmíněno, v jazyce C#. Projekt je založen pro malá zařízení tzv. Smart Device s operačním systémem Pocket PC 2003.

Referenci pro *System.Data.SglServerCe* přidáme v okně *Solution Explorer* pravým kliknutím na *References*. Zdroj dat (data databáze) připojíme pomocí *Add New Data Source*. Jako zdroj dat označíme *My Computer* a cestu k databázi nastavíme na *C:\sqlmobile*.

Tabulky získáme přetažením z okna *Data Source* a zakotvíme je pomocí dokování *Top* a *Buton*.



Obr. 10-2 – Emulátor PocketPC ve Visual Studiu

Knihovna, která zajišťuje veškeré spojení s databází a ovládání synchronizace pro konkrétní programovací jazyk:

C#	using System.Data.SqlServerCe;
Visual Basic	Imports System.Data.SqlServerCe

10.6. Jednotlivé procedury

Procedura pro synchronizaci:

```
private void Sync()
        {
            SqlCeReplication repl = new SqlCeReplication();

            repl.InternetUrl =
@"http://localhost/SQLMobile/sqlcesa30.dll";
            repl.Publisher = @"PetrPC";
            repl.PublisherDatabase = @"SQLMobile";
            repl.PublisherSecurityMode =
SecurityType.NTAuthentication;
            repl.Publication = @"SQLMobile";
            repl.Subscriber = @"sqlmobile";
```

```
            repl.SubscriberConnectionString = @"Data Source='" +
filename + "';Password='';Max Database Size='128';Default Lock
Escalation='100';";
            //repl.AddSubscription(AddOption.CreateDatabase);
            //repl.Synchronize();


            try
            {
                repl.AddSubscription(AddOption.ExistingDatabase);
                //repl.AddSubscription(AddOption.CreateDatabase);
                repl.Synchronize();
            }
            catch (SqlCeException e)
            {
                MessageBox.Show(e.ToString());
            }
        }
```

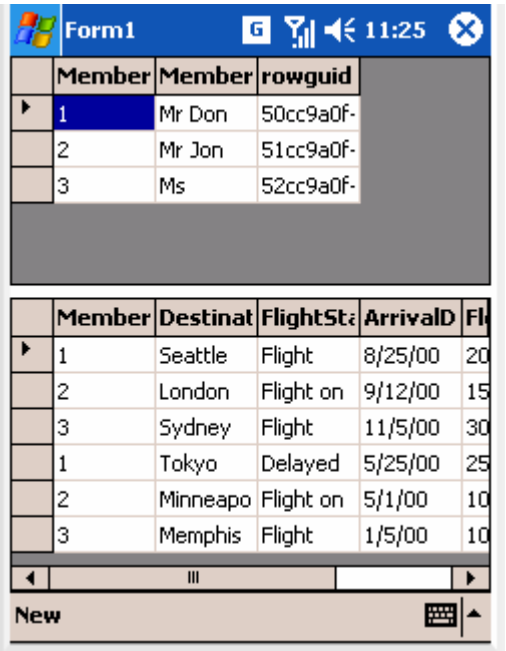
Procedura pro vymazání databáze:

```
private void DeleteDB()
        {
            if (System.IO.File.Exists(filename))
            {
                System.IO.File.Delete(filename);
            }
        }
```

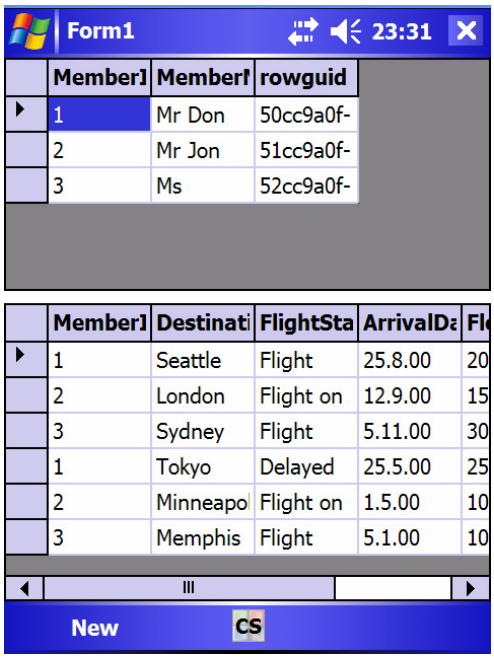
11. Test aplikace

Po přeložení kódu se nainstaluje a spustí aplikace v emulátoru. Program můžeme nahrát i do konkrétního zařízení pomocí programu AktiveSync a synchronizační kolébky.

Na následujících ukázkách programu je vidět, že se nám data synchronizovala s databází vytvořenou v SQL serveru.



Obr. 11-1 – Výsledná aplikace – emulátor



Obr. 11-2 – Výsledná aplikace – PDA C550 Fujistu-Siemens

12. Závěr

Technologie pro vytváření databázových aplikací a aplikací vůbec v jazyku Java pro kapesní počítače PDA zdaleka nejsou na konci svého vývoje. Většina aplikací pro platformu PocekPC, neboli operační systém Windows Mobile, je vytvářena v prostředí .NET kde Java ztrácí na konkurenci. Což je pochopitelné, neboť vše je pod záštitou společnosti Microsoft.

Vzhledem k omezené funkcionalitě, které se podařilo dosáhnout s JDBC, jsme se rozhodli navrhnout a vyzkoušet alternativu v podobě produktu od firmy Microsoft a prostředí .NET. Navržená aplikace byla experimentálně ověřena jak v simulátoru, tak i v reálném PDA C550 od fimy Fujitsu-Siemens.

Seznam použitých zkratek

API	Application Programming Interface
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
CORBA	Common Object Request Broker Architecture
DBMS	Database Management System
GPRS	General Packet Radio Service
GUI	Graphical User Interface
IIS	Internet Information Services
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JDBC	Java Database Connectivity
JVM	Java Virtual Machine
PDA	Personal Digital Assistant
PDAP	Personal Digital Assistant Profile
RMI	Remote Method Invocation
SQL	Structured Query Language
MIDP	Mobile Information Device Profile
WAP	Wireless Application Protocol

tzv.	tak zvaný
např.	například
popř.	popřípadě
resp.	respektive

Seznam použité literatury a zdrojů

[1] Lubomír Brůcha: Java – Hotová řešení, Computer Press 2003, ISBN 80-251-0072-3

[2] Florian Hawlitzek: Java 2 příručka programátora, Grada 2002, ISBN 80-247-9060-2

[3] Pavel Herout: Učebnice jazyka Java, Koop 2000, ISBN 80-7232-115-3

[4] Pavel Herout: Java – grafické uživatelské prostředí a čeština, Koop 2001, ISBN 80-7232-150-1

[5] Internetové stránky Sun: http://java.sun.com/

[6] Internetové stránky www.netbeans.org

[7] Internetové stránky www.interval.cz, Bittnerová, Lucie Rút: Co vás zajímá o J2ME, ale báli jste se zeptat

[8] Internetové stránky www.interval.cz, Jan Šeda: Úvod do JDBC

[9] Internetové stránky http://nb.vse.cz/~zelenyj/, Petr Nejedlý: JDBC

[10] Internetové stránky Rydval.cz, Slávek Rydval: Databáze do dlaně

[11] Internetové stránky http://objekty.pef.czu.cz/2004/sbornik/15_Turecek.pdf

[12] Internetové stránky www.dbsvet.cz, Filip Kinský: Z off-line zařízení na podnikové databáze

[13] Elektronická dokumentace Microsoft SQL Server 2005 Mobile Edition