Microsoft .net

Ľuboslav Lacko

Vývoj aplikácií pre mobilné zariadenia

Microsoft

Vývoj aplikácií pre mobilné zariadenia

Kapitola 1
Úvod, predstavenie a oblasť použitia mobilných zariadení
Servisná kapitola – databázové tabuľky – vytvorenie webových služieb

Kapitola 2
eMbedded Visual Tools 3.0, eMbedded Visual C++ 4.0

Kapitola 3
Compact .NET Framework & SDE
„Connected“ aplikačný scenár I (managed provider pre SQL)
„Connected“ aplikačný scenár II (komunikácia pomocou webových služieb)

Kapitola 4
SQL CE 2.0 (jeho využitie z compact .NET frameworku)
„Disconnected“ aplikačný scenár I (replikace dat SQL <-> SQL CE)

Kapitola 5
Mobile Internet Toolkit (jednoduchá WAP aplikácia)

Kapitola 6
Vývoj pre SmartPhone

Materiály a zdrojové súbory cvičných príkladov sú k dispozícii na adrese
http://msdn.microsoft.cz/docs/BrozuraDevices.zip

Microsoft®

Kapitola 1:

Úvod, predstavenie a oblasť použitia mobilných zariadení
Servisná kapitola – databázové tabuľky – vytvorenie webových služieb

Úvod – oblasť použitia mobilných zariadení

V poslednom čase sa čoraz väčšej popularite tešia prenosné počítače triedy Handheld a Pocket PC. Je to podmienené rozvojom Internetu a mobilnej komunikácie. O osude a úspešnosti veľkých obchodných transakcií často rozhodujú doslova minúty. Každý sa potrebuje operatívne dostať ku svojej elektronickej pošte, prípadne pružne reagovať na rôzne podnety a informácie. Niekedy je potrebné vykonávať diaľkový dohľad nad nejakým technologickým zariadením, ktoré pracuje v bezobslužnom režime. Na prvý pohľad by sa zdalo, že zariadenia tejto triedy budú len „imidžovou hračkou“ v rukách manažérov, prípadne im pomôže organizovať to, s čím každý deň urputne bojujú – čas, prípadne sa na prístrojoch tejto triedy dajú hrať zaujímavé a kvalitné hry, alebo čítať knihy pomocou aplikácie MsReader. Prax ukázala, že s vývojom komunikačných a hardvérových možností týchto zariadení, keď ich procesory sú taktované na stovkách MHz a pamäťová kapacita je minimálne 64 MB sa podstatne rozširuje aj oblasť ich použitia.

Terminológia

Pre získanie orientácie v pojmoch ako operačný systém Windows CE, Windows CE.NET, platforma Pocket PC, Smartphone a podobne zavedieme určité terminologické konvencie. V prvom rade je potrebné rozlišovať medzi operačným systémom a platformou

Operačný systém je hlavný riadiaci program akéhokoľvek počítača. Jeho úlohou je sprístupniť hardvérové porty a zariadenia (displej klávesnica...) jednotlivým aplikáciám a ich prostredníctvom aj používateľom. Je to prvý program, ktorý býva do príslušného počítača zavedený z nejakého pamäťového média pri jeho spustení (proces zavádzania operačného systému sa nazýva aj bootovanie). Obvykle sa operačný systém skladá z jadra a obsluhy zariadení. Typickými príkladmi operačných systémov sú Windows CE, Windows CE .NET, Windows XP, Windows XP Embedded, Windows 98, Windows NT®... Samozrejme nie je žiadny dôvod prečo by v zborníku Microsoftu nemali byť uvedené aj ďalšie populárne operačné systémy ako napríklad MacOS, Linux, prípadne OS/2...

Platforma je vo všeobecnosti definovaná ako určitý hardvér podporovaný množinou programov, modulov, komponentov pre styk používateľom a pochopiteľne aj operačným systémom. Napríklad Pocket PC, Pocket PC 2002, Handheld PC Professional, Handheld PC, Palm-size PC, Auto PC, Smartphone 2002, a Tablet PC. Nevynecháme ani každému známe platformy ako sú desktopy, a notebooky typu PC.

Prehľad platforiem a typov mobilných zariadení

Obrázok 1.1 – Prehľad plaform a typov

Začalo to prístrojmi s klávesnicou. Na tak revolučnú zmenu, ako vreckové počítače bez klávesnice, totiž ešte používatelia neboli pripravení. Prvý počítač triedy Handheld bol predstavený na výstave Comdex v roku 1997. O rok neskôr bol predstavený počítač triedy Handheld Pro. Uhlopriečka monochromatického dotykového displeja bola okolo 6``, prístroje obsahovali typicky 4MB RAM a pracovali pod operačným systémom **Windows CE** 1.0, neskôr vo verzii 2.0. V ďalšom vývoji sa kapacita operačnej pamäti pohla smerom k 32 MB, začali sa používať 256 farebné dotykové displeje. Štandardom bol slot na PCMCIA kartu. Najnovšie handheldy s operačným systémom Windows CE 3.0 disponujú displejmi s rozlíšením 640x240 (polovičné rozlíšenie štandardu VGA), alebo dokonca plným rozlíšením SVGA 640x480. Platforma **Pocket PC** bola vyvíjaná pod kódovým označením Rapier a verejnosti bola predstavená v apríli 2000. A sme v súčasnosti. V októbri 2001 bola uvedená verzia **Pocket PC 2002** (vyvíjaná pod kódovým označením Merlin).

Pocket PC 2002

Ak by sme porovnali grafický návrh používateľského rozhrania medzi staršími operačnými systémami Windows CE a Pocket PC 2002 rozdiel by bol asi ako medzi Windows 98 (Millennium) a Windows XP. Podobne by dopadlo aj porovnanie zapuzdrených technológií, kde rozdiely na prvý pohľad nie sú vidieť, o to sú však významnejšie. Nový operačný systém je graficky aj farebne oveľa nápaditejšie vyriešený, čo nepochybne prispeje k lepšiemu ovládaniu a pohodliu používateľa. Nebudeme popisovať klasické funkcie ako sú kalendár, plánovač, adresár, programy balíka Pocket Office a podobne, tieto aplikácie sú známe z predchádzajúcich verzií Windows CE. Nová verzia operačného systému od Microsoftu totiž poskytuje používateľom aj oveľa viac technologických možností, hlavne čo sa týka bezdrôtového prepojenia, od lokálnych sietí (802.11b) a osobných sietí (Bluetooth) po diaľkové siete (cez CDPD, CDMA, GSM alebo prostredníctvom dvojstranných riešení s využitím existujúcich operátorských sietí).

Obrázok 1.2 – Pocket PC 2002 – menu START

Softvér dodávaný s platformou Pocket PC 2002 zahrňuje aj novú technológiu Server ActiveSync®, ktorá umožňuje priamu serverovú synchronizáciu elektronickej pošty, kalendára a kontaktov medzi zariadeniami na báze Pocket PC 2002 a servermi Exchange 2000. Server ActiveSync je dodávaný ako komponent **Microsoft Mobile Information 2002 Server,** a je navrhnutý tak, aby podporoval káblové aj bezdrôtové prepojenie. Pre získanie určitého prehľadu by sme mali aspoň stručne predstaviť niektoré prístroje. Pretože modely jednotlivých výrobcov sa od seba príliš nelíšia, predstavíme ako typického predstaviteľa modely Compaq iPAQ. Po fúzii HP a Compaq to bude práve rada iPAQ, v ktorej bude HP pokračovať.

Compaq iPAQ rady 3600 a 3800

Dominantným prvkom matne strieborných prístrojov je farebný podsvietený dotykový displej a elipsovité kurzorové tlačidlo. Okrem neho sú na prednom paneli ešte štyri malé tlačidlá, ktorých funkcia sa dá preprogramovať, vypínacie tlačidlo a na pravej strane je „pohotovostné“ tlačidlo pre záznam zvuku. V hornej časti pod tmavým plexisklom je reproduktor, čidlá pre IrDa prenos a u typovej rady 3800 aj zásuvné miesto pre pamäťovú kartu. Ďalšiu PCMCIA (alebo CF) kartu je možné umiestniť do špeciálneho prípravku (nutné dokúpiť), ktorý sa na prístroj nasúva zozadu. Prístroje typového radu IPAQ 3600 majú reproduktor v telese veľkoplošného tlačidla na ovládanie kurzora. Najdôležitejším ovládacím prvkom prístroja je samozrejme dotykový displej, ovládaný pomocou pera. Pero je len pasívny dotykový prvok, aktívny je displej.

Technické údaje prístroja Compaq iPAQ 3870

Procesor: 206 MHz Intel strong ARM 32-bit
Pamäť: ROM 64 MB, RAM 32 MB
Displej: HR/TFT LCD 240 x 320, 65 000 farieb, prispôsobenie jasu svetelným podmienkam okolia
Externá pamäť: Slot na pamäťovú kartu SD a konektivita pomocou násuvnej „vestičky“
Batéria: typ Lithium Polymer – kapacita na približne 14 hodín prevádzky
Hmotnosť: 190g
Dodávaný softvér: iPAQ Task Manager, IBM via voice command and control, iPresenter Powerpoint Converter for Pocket PC, Dashboard Encryption, Java VM, eWallet, Sega Game Pack, Vegas Game Pack, iPAQ Reference Guide...
Komunikačné možnosti: Integrovaná technológia Bluetooth (rada 3800), pripojenie pomocou mini docking station prostredníctvom sériového, alebo USB rozhrania, IrDA port s rýchlosťou 115kbs

Bezdrôtové prepojenie PDA v sieti

Každý kto sa pozrie na vreckový počítač uväznený v „mini-docking station“ a pripojený k počítaču káblom si uvedomí, že toto nie je to pravé riešenie, ktoré by zaujalo svojou portabilitou. Preto je jasné, že v tejto oblasti nájdu uplatnenie hlavne bezdrôtové komunikačné technológie Bluetooth a Wireless LAN.

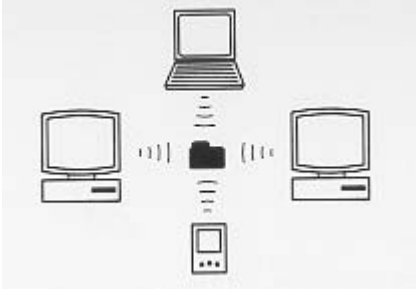
Bluetooth™ Je názov komunikačnej technológie definovanej konzorciom firiem pre bezdrôtovú komunikáciu medzi zariadeniami na krátku vzdialenosť. Bluetooth sa využíva pre stolné počítače, prenosné počítače, personálne digitálne asistenty (PDA), počítačové periférie, mobilné telefóny a iné domáce a technologické zariadenia. Komunikácia prebieha duplexne vo voľnom kmitočtovom pásme 2.45 GHz s dosahom 10 až 15 metrov pri prenosovej rýchlosti 712 Kbps až 2Mbps.

Wireless LAN Lokálna počítačová sieť LAN spája dva, alebo viac počítačov a umožňuje zdieľať disky, súbory, periférie, alebo sieťové služby. Obvykle je tvorená počítačmi poprepájaných viac, alebo menej štruktúrovanou kabelážou. Ak si predstavíme mobilné zariadenia, už pri vyslovení slova „kábel" priam cítime, ako sa táto mobilita vytráca. Preto mobilné zariadenia idú ruka v ruke s bezdrôtovými sieťami. Len takto môže byť zaistená skutočná mobilita. Pri bezdrôtových sieťach sú informácie prenášané éterom, v tomto prípade prostredníctvom rádiového prenosu v pásme 2.4 GHz rýchlosťou 11 Mbps.

Architektúra bezdrôtových sietí

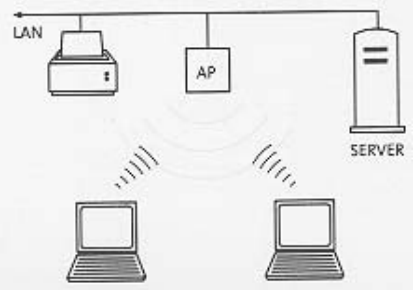
Pri inštalácii sieťovej vrstvy si môžeme vybrať dva základné typy použitej sieťovej architektúry:
- Ad-hoc Networking
- Infrastructure Networking

Ad-hoc Networking taktiež nazývaná **Peer-to-peer** je určená pre lokálne siete s menším počtom počítačov. Umožňuje zdieľanie diskov, súborov, tlačiarní a prístup na Internet cez zdieľaný modem. Výhodou je jednoduchá inštalácia a nízke náklady



Obrázok 1.3 – Architektúra Peer-toPeer

Infrastructure Networking Pri tejto architektúre, na rozdiel od siete nastavenej ako Ad-hoc Networking, kde je priamy tok údajov ku každému počítaču, údaje prechádzajú cez takzvaný Access Point. Tento môže byť hardvérový alebo softvérový, vytvorený pomocou počítača PC s wireless kartou a príslušným softvérom.

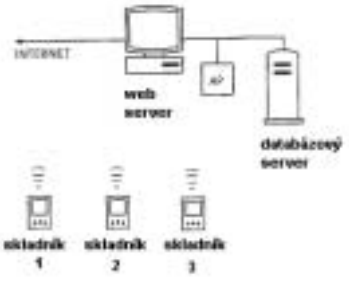


Obrázok 1.4 – Infrastructure Networking

Access Point obvykle umožňuje aj prístup do klasických „káblových" sietí. Výhodou je zvýšenie dosahu, pretože každú účastník rádiovej siete má spojenie v dosahu Access Pointu. Nevýhodou sú vyššie náklady, zahrňujúce aj hardvérový, alebo softvérový Access Point. Základným problémom všetkých rádiových sietí je **bezpečnosť údajov**. Aby sa zabránilo úniku informácií, ktoré sú voľne v éteri, napríklad odposluchom mimo budovy, je možné údaje šifrovať pomocou systému **Wired Equivalent Privacy – WEP**. WEP používa k šifrovaniu údajov 40 bitový kľúče, ktoré byť identické u všetkých klientov siete.

Príklad použitia rádiovej siete.

Typický príklad použitia rádiovej siete je aplikácia typu e-business. Predstavme si napríklad zásielkový predaj tovaru. Okrem dátových transakcií je potrebné tovar v sklade nájsť a vyskladniť. Ak v budove skladu zriadime rádiovú sieť, môžeme priamo pomocou PDA úkolovať jednotlivých skladníkov a kontrolovať potvrdenie vyskladnenia tovaru.



Obrázok 1.5 – Príklad použitia WLAN

Komunikačný program Microsoft ActiveSync

Pre komunikáciu či už prostredníctvom sériového alebo USB portu alebo pre bezdrôtovú komunikáciu je potrebné nainštalovať do desktopu alebo notebooku komunikačný a synchronizačný softvér Microsoft ActiveSync. Počas inštalácie nastavíme, ktoré položky programu MS Outlook chceme synchronizovať s PDA. Po nainštalovaní sa synchronizačný program automaticky spustí a zaháji sa synchronizácia. O správnej činnosti svedčí okrúhla zelená ikonka v pravom dolnom rohu.
Ak chceme snímať obrazovky z PDA, napríklad z dôvodu výroby dokumentácie k vyvíjanému projektu, alebo pre pripojenie k projektoru a podobne ideálny je pre tento účel program **Microsoft Remote Display Control**.

Trocha histórie okolo možností vývoja aplikácií

Mobilné zariadenia majú pomerne všestranné využitie aj so zabudovaným softvérom. Okrem základných utilít operačného systému obvykle obsahujú aj základné programy pre personálny manažment, textový procesor, tabuľkový kalkulátor, prehliadač webových stránok... Niekedy však ani toto programové vybavenie k našej práci nepostačuje, a v takom prípade sme nútení zaobstarať si vhodný softvér, alebo vyvinúť pre tieto zariadenia softvér vlastný. Jednoduché programy pre mobilné zariadenia môžeme v niektorých ojedinelých prípadoch vyvíjať priamo na konkrétnom type vreckového počítača, na ktorom potom program aj pobeží, napríklad v niektorom dialekte programovacieho jazyka BASIC. V drvivej väčšine však programy pre mobilné zariadenia vyvíjame na inej platforme, napríklad na klasickom PC.

Ak nepočítame programovateľné kalkulátory a diáre, prvými modelmi vreckových počítačov, pre ktoré bolo možné vyvíjať programy boli palmtopy, napríklad **Hewlett Packard 200 LX** a **PSION 3A**.

Hewlett Packard 200 LX si rýchlo získal obľubu pre jeho kompatibilitu z vtedy najrozšírenejším operačným systémom MS-DOS. Bol postavený na procesore NEC kompatibilnom s procesorom Intel x86. Mal 1 MB pamäti RAM, ktorá sa dala rozdeliť medzi pamäť pre operačný systém a RAM-DISK. V pamäti ROM bol uložený samotný operačný systém, v tomto prípade trocha okliešteny MS-DOS 3.0 a firemný softvér pre personálny manažment. Prístroj mal sériový port, IrDA rozhranie a jeden slot pre PCMCIA kartu. S programovým vybavením pre tento model a ani s jeho vývojom preto nebol nijaký problém. Pokiaľ tomu nebránila veľkosť pamäti, alebo rozlíšenie grafiky (CGA 640x200) fungovali na ňom všetky programy pre MS-DOS vrátane populárneho Norton Commandera a v našich končinách populárneho textového editora s diakritikou T602. Keďže HP 200 LX mal procesor kompatibilný s procesorom Intel 8086, bolo možné pre vývoj aplikácií použiť ľubovoľný assembler, alebo vyšší programovací jazyk pre tento procesor a pre operačný systém MS-DOS. Po nainštalovaní jednoduchého interpretera programovacieho jazyka BASIC bolo možné vytvoriť aj aplikáciu priamo na vreckovom počítači. Šťastný majiteľ pamäťovej karty si mohol nainštalovať aj kompilátory vyšších programovacích jazykov, napríklad jazyka C

PSION série 3A. Tento model mal vlastný operačný systém, ktorý obsahoval vyspelý firemný softvér pre personálny manažment PIM. Do prístroja bolo možné zasunúť dve FLASH karty a dodávala sa k nemu aj plná lokalizácia do češtiny a slovenčiny. Vyvíjať aplikácie pre model **PSION 3A**, hlavne na platforme PC bolo o niečo zložitejšie, naopak vývoj aplikácií priamo na vreckovom počítači bol pomerne jednoduchý, pretože počítač mal vlastný operačný systém so vstavaným programovacím jazykom OPL, čo bol jazyk vzdialene podobný BASICu. Okrem jazyka OPL bolo možné vyvíjať aplikácie pre tento prístroj vo vývojovom prostredí SIBO pod operačným systémom MS-DOS. Tento balík obsahoval výkonný kompilátor jazyka C (bohužiaľ neobjektový) Speed C. Po nainštalovaní knižníc SIBO bolo možné vyvíjať kvalitné aplikácie, ktoré mohli využiť všetky výhody multitaskingového operačného systému a vstavaných periférií. Čo sa týka kompatibility, firma PSION prešla s novým modelom **PSION séria 5** na operačný systém EPOC a bolo nutné kúpiť nové vývojové prostredie pre tento operačný systém. Toto prostredie sa dodávalo ako doplnok k známemu vývojovému prostriedku Microsoft Visual Basic.

Windows CE. Oveľa nádejnejšia začala byť situácia s kompatibilitou u nového operačného systému Microsoft Windows CE vo verziách 1.0 a 2.0. Síce nebolo možné používať programy napísané pre Windows 95, NT, ani programy pre DOS, takže bolo nutné vyvíjať programové vybavenie pre túto triedu počítačov osobitne. Situáciu naviac komplikovala skutočnosť, že sa v tejto triede počítačov používalo niekoľko nekompatibilných procesorov (MIPS, SH3, SH4, ARM...). Našťastie bola zachovaná aspoň čiastočná kompatibilita na úrovni zdrojových kódov. Microsoft dodával softvérový balík pre Visual Basic a Visual C++. Neskôr sa objavil Microsoft Windows CE toolkit for Visual C++.

Súčastný stav a blízka budúcnosť

Aj momentálny stav u Microsoftu, čo sa týka vývojových nástrojov pre mobilné zariadenia je pomerne zaujímavý. Nie, že by nebolo po čom siahnuť, to rozhodne nie. Ponuka je bohatá a skôr je problémom nájsť ten správny nástroj pre príslušný typ aplikácie. Na výber máme vývojové nástroje **Microsoft® eMbedded Visual Tools 3.0, Microsoft® eMbedded Visual C++ 4.0, Microsoft Mobile Internet Toolkit (MMIT),** a **Smart Device Extensions (SDE)**. Posledne menovaný nástroj sa inštaluje ako doplnok do vývojového prostredia **Visual Studio .NET**. Významné zjednotenie a zapúzdrenie niektorých zo spomínaných technológií prinesie nová verzia tohoto vývojového prostredia. V dobe písania zborníka bola už k dispozícii stabilná Alfa verzia nového vývojového prostredia **Visual Studio .NET 2003** s kódovým názvom Everett s integrovanou technológiou Smart Device Extensions. Taktiež aj balík Microsoft Mobile Internet Toolkit bol do tejto verzie integrovaný pod názvom **ASP.NET Mobile Controls**

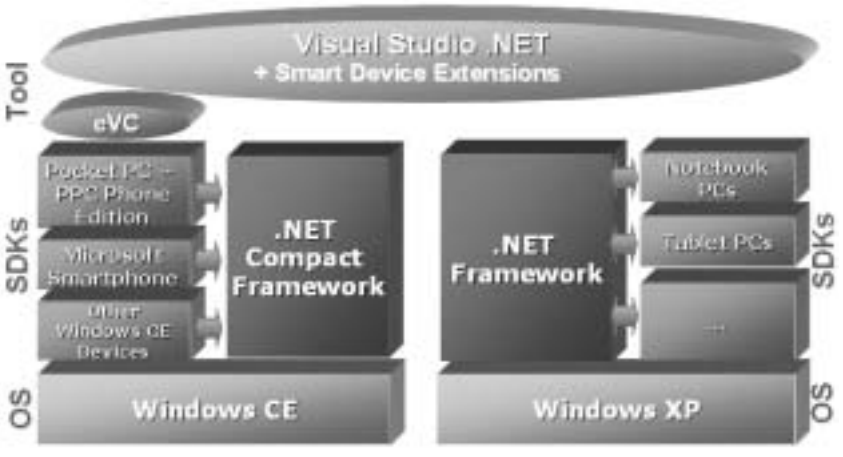
Súčastná verzia vývojového prostredie **Visual Studio .NET** je k dispozícii v troch variantách:
- **Architekt**
- **Developer**
- **Professional**

Obrázok 1.6 – Verzie vývojového prostredia Visual Studio .NET

Stručný prehľad základných vlastností jednotlivých verzií je v tabuľke.

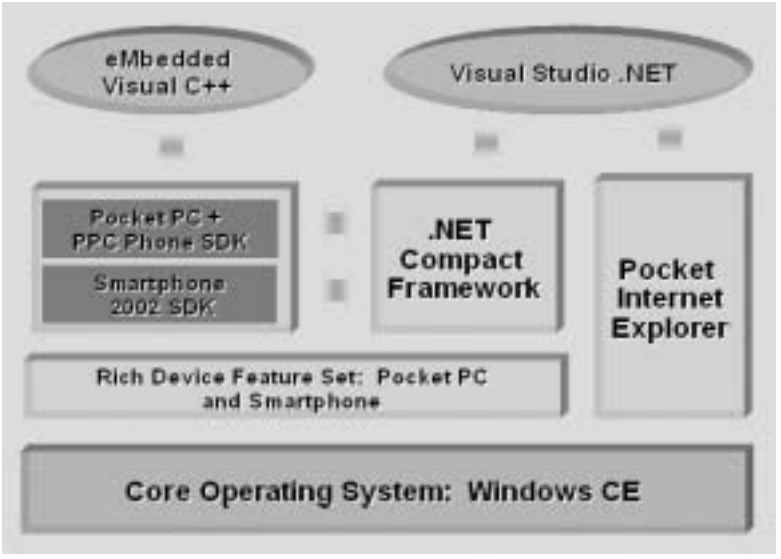
Visual Studio.NET – prehľad vlastností jednotlivých verzí			
	Architekt	**Developer**	**Professional**
Visual Studio.NET IDE	•	•	•
Programovacie jazyky Visual Basic,C#, C++	•	•	•
Microsoft Visual SourceSafe	•	•	
Testovanie výkonu a funkčnosti webových XML služieb	•	•	
Použitie „Enterprise frameworks“	•	•	
Microsoft Windows Server	•	•	
Microsoft SQL Server	•	•	
Microsoft Exchange Server	•	•	
Microsoft Commerce Server	•	•	
Microsoft Host Integration Server	•	•	
Microsoft BizTalk Server	•		
Tvorba „Enterprise frameworks“	•		
Dátové modelovanie, reverzné inžinierstvo	•		
Modelovanie obchodných procesov pomocou UML	•		
Generovanie programov Visual Basic,C#, C++, ...	•		

Vývojové prostredie **Visual Studio.NET** doplnené o balík **Smart Device Extensions** zastrešuje vývoj aplikácií pre desktopy aj pre mobilné zariadenia. Azda najlepšie je možné charakterizovať súčastný stav nasledujúcou schémou



Obrázok 1.7 – Prehľad nástrojov a možností pre vývoj aplikácií

Už na prvý pohľad to vyzerá nádejne. Pomocou jediného vývojového prostredia, samozrejme pri rešpektovaní rozdielov medzi .NET Framework a .NET Compact Framework môžeme vyvíjať aplikácie jednak pre počítače triedy PC a taktiež aj pre mobilné zariadenia. Ešte názornejšie demonštruje rozdiely medzi vývojovým prostredím typu eMbedded a balíkmi SDE a MMIT táto schéma



Obrázok 1.8 – Prehľad nástrojov a možností pre vývoj aplikácií

Vývojové nástroje **eMbedded Visual Tools 3.0** a **eMbedded Visual C++ 4.0** sú od Visual Studia.NET úplne nezávislé. Naznačme oblasť použitia jednotlivých vývojových nástrojov.

Microsoft® eMbedded Visual C++ 3.0

- Ovládače pre Pocket PC a iné mobilné zariadenia
- Natívne aplikácie pre mobilné zariadenia
- Aplikácie u ktorých je priorita rýchlosť
- Hry a iné aplikácie, ktoré vyžadujú rýchlu grafiku
- COM servery a Active X komponenty

Microsoft® eMbedded Visual Basic 3.0

- Aplikácie u ktorých je prioritná krátka doba vývoja
- Aplikácie typu user – interface zložené prevažne z komponentov
- Tvorba prototypov RAD (Rapid Application development) aplikácií
- Tvorba jednoduchých utilít

Microsoft® eMbedded Visual C++ 4.0 (zmeny v porovnaní s eMbedded Visual C++ 3.0)

- Vývoj aplikácií pre Windows CE .NET
- Vylepšené ladenie aplikácií JIT (Just – In – Time), napríklad pre odhaľovanie neošetrených výnimiek
- Vylepšený softvérový emulátor

Visual Studio.NET doplnené o Smart Device Extensions (SDE)

- Aplikácie využívajúce XML a webové služby
- Aplikácie, ktoré musia pracovať v pripojenom aj odpojenom režime
- Vývoj prípadne migrácia aplikácii v programovacích jazykoch Visual Basic a Visual C#
- Vývoj aplikácie, ktorá bude nasadená v heterogénnom prostredí, napríklad na desktope, serveri aj mobilnom zariadení, pričom sa samozrejme predpokladajú rôzne procesory a rôzne verzie operačného systému Windows
- Aplikácie vyžadujúce spoľahlivé a bezpečné prostredie

Microsoft Mobile Internet Toolkit (MMIT) v novej verzii ASP.NET Mobile Controls

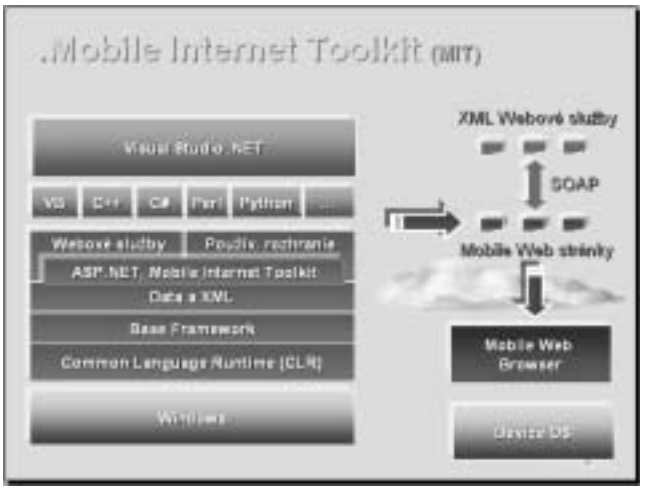
- Ťažisko aplikácie je na serveri, z mobilného zariadenia sa k nej pristupuje pomocou webového prehliadača
- MMIT je založené na technológii ASP.NET stránok, HTML, cHTML (Compact HTML) a WML (Wireless Markup Language)

Možno viac, než podrobný popis nám oblasť použitia jednotlivých nástrojov naznačí prehľadná tabuľka

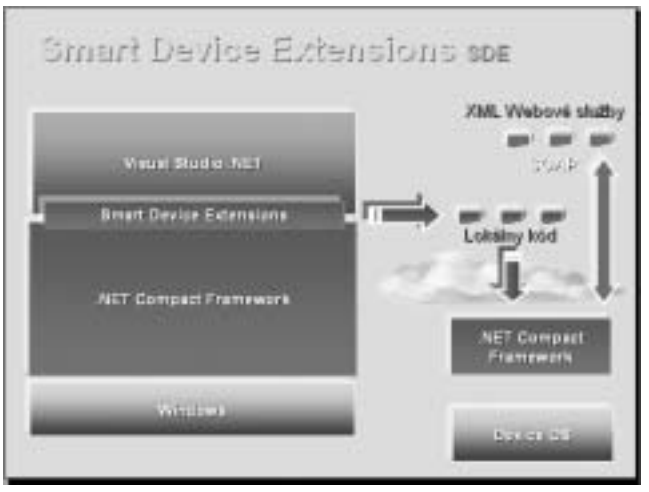
	eMbedded Visual Tools 3.0		**Windows CE .NET + Platform Builder**	**Visual Studio .NET** v novej verzii Everett platí vyšrafovaný stĺpec ako celok		
	eMbedded Visual C 3.0	**eMbedded Visual Basic 3.0**	**eMbedded Visual C 4.0**	**SDE**	**MMIT**	**VS .NET**
Native Code	X		X			X
Managed Code		(X) Nie na báze .NET		X		X
Pocket PC Pocket PC 2002	X	X		X	X	X SDE, MMIT
Smartphone 2002	X			+	+	X SDE, MMIT
Non-Microsoft Mobile Devices					(X) potrebný prehliadač	X SDE, MMIT
Embedded Devices	X	(X) Musí byť predinštalovaný VB Runtime	X	(X) Len Windows CE .NET	X	X len XP Embedded
COM Development	X	(X) nie je možné vyvíjať ActiveX	X			X
Driver Development	X		X			X
Debug						
Drivers	X		X			X

X Úplná podpora
(X) Podpora s určitými obmedzeniami+ v aktuálnej verzii nie je podporované, podpora sa predpokladá v ďalších verziách

Podrobnejšie vysvetlenie si určite zaslúži rozdiel medzi **Microsoft Mobile Internet Toolkit (MMIT)** a **Smart Device Extensions (SDE)** z hľadiska technológie a zaradenia v rámci Visual Studia .NET.



Obrázok 1.9 – Microsoft Mobile Internet Toolkit



Obrázok 1.10 – Smart Device Extensions

Ako nakoniec všetko na tejto planéte, majú aj technológie MMIT a SDE svoje výhody a nevýhody

Microsoft Mobile Internet Toolkit (MMIT)

Výhody

- Podpora najširšieho spektra zariadení
- Prístup prostredníctvom browsera

Nevýhody

- aplikácie nepracujú v off - line režime
- Aplikácie málo využívajú výkon lokálneho zariadenia
- Obmedzenia týkajúce sa používateľského rozhrania

Smart Device Extensions (SDE)

Výhody

- Aplikácie pracujú v obidvoch režimoch off - line aj on -line
- Plné využitie možností poskytovanými platformou Pocket PC
- Spolupráca s SQL Server™ CE
- Využitie lokálneho procesora, grafiky, multimediálnych možností

Nevýhody

- Množstvo podporovaných platforiem a zariadení je obmedzené

Servisná kapitola – Vytvorenie webovej služby

V publikácii budeme pracovať s údajmi v databázach a využívať webové služby. Preto nezaškodí zoznámiť sa s cvičnými tabuľkami a ukázať postup vytvorenia jednoduchej webovej služby.

Cvičné údaje

Vo viacerých prípadoch budeme pracovať s údajmi v databáze pod správou databázového servera SQL Server 2000. Spolu s databázovým serverom sa nainštaluje cvičná databáza **Northwind**. V tejto databáze budeme pracovať s databázovou tabuľkou **Employees,** ktorá obsahuje údaje o zamestnancoch firmy. Jej návrhová štruktúra je nasledovná:

EmployeeID	int
LastName	nvarchar (20)
FirstName	nvarchar (10)
Title	nvarchar (30)
TitleOfCourtesy	nvarchar
BirthDate	datetime
HireDate	datetime
Address	nvarchar (60)

```
City            nvarchar (15)
Region          nvarchar (15)
PostalCode      nvarchar (10)
Country         nvarchar (15)
HomePhone       nvarchar (24)
Extension       nvarchar (4)
Photo           image
Notes           ntext
ReportsTo       int
PhotoPath       nvarchar (255)
```

Tabuľka obsahuje 9 záznamov a pretože niektoré z nich sú pomerne rozsiahle (napríklad stĺpec Notes) a foto sa v textovej podobe ani jednoducho vyjadriť nedá vypíšeme len obsah niekoľkých stĺpcov, napríklad príkazom

```
SELECT LastName, FirstName, Title, Address, City, Country
FROM Employees
```

LastName	FirstName	Title	Address	City	Country
Davolio	Nancy	Sales Representative	507 - 20th Ave. E. Apt. 2A	Seattle	USA
Fuller	Andrew	Vice President, Sales	908 W. Capital Way	Tacoma	USA
Leverling	Janet	Sales Representative	722 Moss Bay Blvd.	Kirkland	USA
Peacock	Margaret	Sales Representative	4110 Old Redmond Rd.	Redmond	USA
Buchanan	Steven	Sales Manager	14 Garrett Hill	London	UK
Suyama	Michael	Sales Representative	Coventry House Miner Rd.	London	UK
King	Robert	Sales Representative	Edgeham Hollow Winchester Way	London	UK
Callahan	Laura	Inside Sales Coordinator	4726 - 11th Ave. N.E.	Seattle	USA
Dodsworth	Anne	Sales Representative	7 Houndstooth Rd.	London	UK

```
(9 row(s) affected)
```

Pre jednoduchšie demonštrácie budeme používať tabuľku **Region,** ktorá má len dva stĺpce

```
RegionID                int
RegionDescription       nvchar 50)
```

Túto tabuľku môžeme pokojne vypísať celú SQL príkazom: **`select *from region;`**

RegionID	RegionDescription
1	Eastern
2	Western
3	Northern
4	Southern

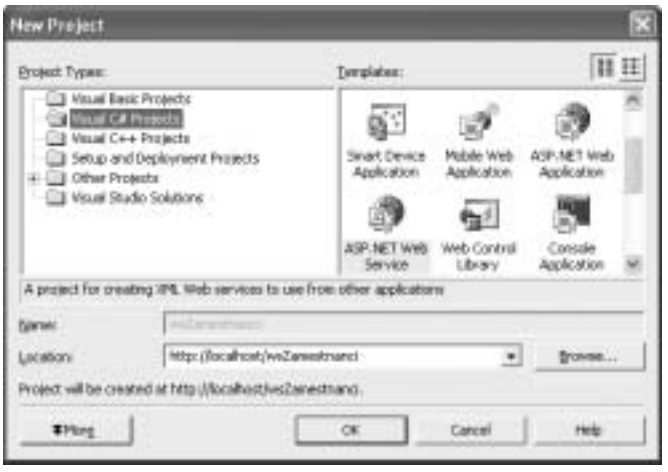
(4 row(s) affected)

Vytvorenie webovej služby využívajúcej databázu (Visual C#)

Vo viacerých cvičných príkladoch budeme využívať webovú službu. Preto si jednu takúto jednoduchú službu vytvoríme. Jej úloha bude jednoduchá. Po zadaní ID zamestnanca (stĺpec **`EmployeeID`** z tabuľky Employees databázy Northwind) vráti údaje o príslušnom zamestnancovi

Krok 1. – Vytvorenie aplikácie typu Smart Device Application

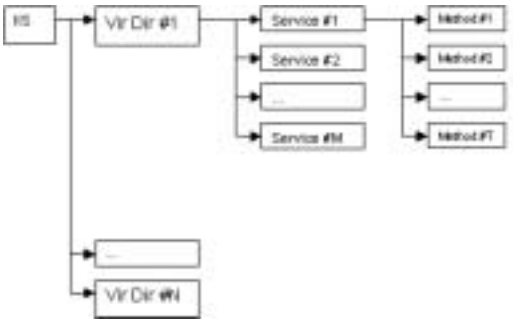
Pomocou menu **File | New | Project** vytvoríme nový projekt typu **ASP.NET Web Service**. Projekt vytvoríme v zložke **Visual C# Projects**. Napokon v úvodnom dialógu pomenujeme projekt ako **wsZamestnanci**.



Obrázok 1.11 – Vytvorenie projektu typu webová služba

Webová služba bude prístupná cez adresu http://localhost/wsZamestnanci a štandardne bude v adresári ***c:\inetpub\wwwroot\wsZamestnanci***.

Hierarchická štruktúra IIS – virtuálne adresáre, služby a ich metódy je vyjadrená schémou:



Obrázok 1.12 – Hierarchická štruktúra prístupu k webovým službám a ich metódam

Na úvodnej obrazovke Visual Studia klikneme na link **click here to switch to code view**. V tomto prípade nás sprievodca vytvorením aplikácie nenechal trápiť, a rovno nám ukázal ako sa taká služba programuje. Takže stačí odstrániť komentáre na začiatku riadkov a prvú webovú službu HelloWorld() máme hotovú

```
// WEB SERVICE EXAMPLE
   // The HelloWorld() example service returns the string Hello World
   // To build, uncomment the following lines then save and build the project
   // To test this web service, press F5

     [WebMethod]
     public string HelloWorld()
     {
       return "Hello World";
     }
```

Pomocou menu **Build | Build wsZamestnanci** aplikáciu preložíme a zostavíme. Pomocou menu **Debug | start** môžeme webovú službu vyskúšať.

Obrázok 1.13 – Testovanie webovej služby

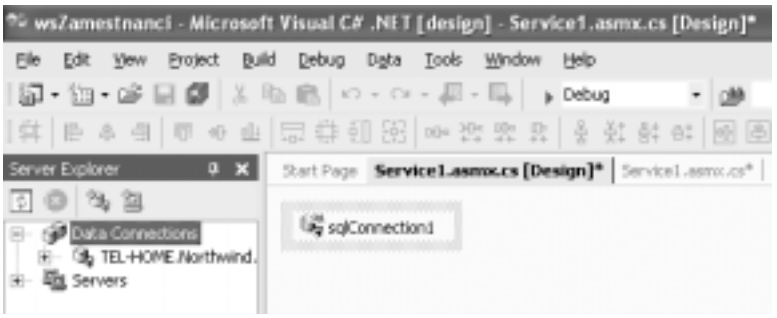
Po zatlačení tlačidla Invoke sa zobrazia údaje poskytnuté webovou službou, v tomto prípade reťazec **Hello World** vo formáte XML.

<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="**http://tempuri.org/**">**Hello World<**/string>

Po tomto precvičení vytvoríme ďalšiu metódu, ktorá nám na základe ID vráti údaje o zamestnancovi. Najskôr však musíme vytvoriť prepojenie na SQL Server. Ľavé okno prepneme do režimu **Server Explorer** a vytvoríme nové pripojenie v zložke **Data Connections**. (pravé tlačidlo, voľba **Add Connection**)

Obrázok 1.14 – Vytvorenie prepojenia na databázu

Po vytvorení pripojenia, toto kurzorom myši presunieme na plochu aplikácie.



Obrázok 1.15 – Vytvorenie prepojenia na databázu

Teraz už môžeme pristúpiť k naprogramovaniu aplikačnej logiky. V zložke Service1asmx.cs si všimnime kód, ktorý bol pridaný počas presunu komponenty sqlConnection.

```
private System.Data.SqlClient.SqlConnection sqlConnection1;
```

Najskôr pridáme na začiatok súboru kód

```
using System.Data.SqlClient;
```

a na koniec súboru metódu **DajZamestnanca(**

```
[WebMethod]
public DataSet DajZamestnanca(int id)
{
    string strSQL = "SELECT LastName, FirstName, Title, Address, City, Country, Notes FROM
Employees where EmployeeID=" + id.ToString();
    SqlDataAdapter da = new SqlDataAdapter(strSQL, sqlConnection1);
    DataSet ds = new DataSet();
    da.Fill(ds, "Employees");
    sqlConnection1.Close();
    return ds;
}
```

Metóda je v poriadku, bez problémov funguje, ale má to celé jeden háčik a to je bezpečnosť. V prípade ak je parametrom číslo to príliš nevadí, no ak by bol parametrom reťazec, mohla by vzniknúť pomerne nebezpečná bezpečnostná diera, nakoľko pri troche šikovnosti by bolo možné vytvoriť vnorený dotaz obsahujúci napríklad sekvenciu „**...Delete FROM Employees...**“. Preto je lepšie vytvoriť SQL príkaz cez parametre

```
[WebMethod]
public DataSet DajZamestnanca(int id)
{
    SqlCommand cmd = new SqlCommand("SELECT LastName, FirstName, Title, Address, City, Country,
Notes FROM Employees where EmployeeID = @EMPID", sqlConnection1);
    cmd.Parameters.Add("@EMPID", id);
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    DataSet ds = new DataSet();
    da.Fill(ds, "Employees");
    sqlConnection1.Close();
    return ds;
}
```

Pomocou menu **Debug | start** môžeme novú metódu webovej služby vyskúšať.

Obrázok 1.16 – Testovanie webovej služby

Všimnime si doporučenie aby sme zmenili adresu namespace http://tmpuri.org/ na nejakú inú, ktorá bude unikátna pre každú konkrétnu aplikáciu. V tomto zborníku použijeme napríklad (fiktívnu) adresu http://lubolacko.sk/brozury/smartdevices/v1/.

Urobme tak a pred riadok

```
public class Service1 : System.Web.Services.WebService
{
```

dopíšme riadok

```
[WebService(Namespace="http://lubolacko.sk/brozury/smartdevices/v1/")]
```

Po zatlačení tlačidla **Invoke** sa aj v tomto prípade zobrazia údaje poskytnuté webovou službou, v tomto prípade údaje o príslušnom zamestnancovi vo formáte XML.

```
<?xml version="1.0" encoding="utf-8" ?>
<DataSet xmlns="http://lubolacko.sk/brozury/smartdevices/v1/">
<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="sk-SK">
    <xs:complexType> <xs:choice maxOccurs="unbounded"> <xs:element name="Employees">
        <xs:complexType> <xs:sequence>
            <xs:element name="LastName" type="xs:string" minOccurs="0" />
        <xs:element name="FirstName" type="xs:string" minOccurs="0" />
        <xs:element name="Title" type="xs:string" minOccurs="0" />
            <xs:element name="Address" type="xs:string" minOccurs="0" />
            <xs:element name="City" type="xs:string" minOccurs="0" />
            <xs:element name="Country" type="xs:string" minOccurs="0" />
            <xs:element name="Notes" type="xs:string" minOccurs="0" />
</xs:sequence></xs:complexType></xs:element></xs:choice></xs:complexType>
</xs:element>
  </xs:schema>
- <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
- <NewDataSet xmlns="">
- <Employees diffgr:id="Employees1" msdata:rowOrder="0">
```

<LastName>**Leverling**</LastName>
<FirstName>**Janet**</FirstName>
<Title>**Sales Representative**</Title>
<Address>**722 Moss Bay Blvd.**</Address>
<City>**Kirkland**</City>
<Country>**USA**</Country>
<Notes>J**anet has a BS degree in chemistry from Boston College (1984). She has also completed a certificate program in food retailing management. Janet was hired as a sales associate in 1991 and promoted to sales representative in February 1992.**</Notes>
</Employees></NewDataSet></diffgr:diffgram></DataSet>

Sú to síce viac údaje pre počítač a ten k nim bude prostredníctvom volania metód webovej služby aj pristupovať, ale pri trochu skúseností sa tieto údaje prečítať aj priamo človekom. Podobne môžeme pridať aj ďalšie metódu webovej služby, napríklad pre vrátenie názvu regiónu, na základe zadaného ID.

```
[WebMethod]
public DataSet DajRegion(int id)
{
    SqlCommand cmdx = new SqlCommand("SELECT * FROM Region where RegionID = @REGID",
sqlConnection1);
    cmdx.Parameters.Add("@REGID", id);
    SqlDataAdapter da = new SqlDataAdapter(cmdx);
    DataSet ds = new DataSet();
    da.Fill(ds, "Regions");
    sqlConnection1.Close();
    return ds;
}
```

alebo všetkých zákazníkov, ktorí bývajú v zadanom meste

```
[WebMethod]
public DataSet DajZakaznikovZmesta(string sCity)
{
    SqlCommand cmd = new SqlCommand("SELECT ContactName, Address FROM Customers WHERE City=
@PCITY", sqlConnection1);
    cmd.Parameters.Add("@PCITY", sCity);
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    DataSet ds = new DataSet();
    da.Fill(ds, "Employees");
    sqlConnection1.Close();
    return ds;
}
```

Vo všetkých metódach sme z bezpečnostných dôvodov používali pre konštrukciu objekt `SqlCommand`.

!! Bezpečnostné riziko !!

V snahe po univerzálnosti nie je problém napísať takú metódu webovej služby, ktorá vráti XML dataset z akejkoľvek tabuľky, alebo kombinácie tabuliek databázy Northwind na základe SQL dotazu.

```
[WebMethod]
public DataSet DotazNW(string strSQL)
{
    SqlDataAdapter da = new SqlDataAdapter(strSQL, sqlConnection1);
    DataSet ds = new DataSet();
    da.Fill(ds, "Northwind");
    sqlConnection1.Close();
    return ds;
}
```

Zdanlivo krásne a všeobecne použiteľné. Áno všeobecne použiteľné dokonca aj na vymazanie údajov. Ukážme si to na jednoduchom príklade. Aby sme nepoškodili tabuľky cvičnej databázy Northwind, vytvorme si v tejto cvičnej databáze jednoduchú tabuľku pomocou konzolovej aplikácie Query Analyser a naplňme ju niekoľkými údajmi.

```
CREATE TABLE tovar
(
     nazov      VARCHAR(25),
     cena       DECIMAL(9,2)
);

INSERT INTO tovar VALUES('Banany','39.90');
INSERT INTO tovar VALUES('Mrkva','13.30');
```

O jej obsahu sa môžeme presvedčiť SQL príkazom SELECT v konzolovej aplikácii

```
select * from tovar;

nazov          cena
Banany         39.90
Mrkva          13.30

(2 row(s) affected)
```

K rovnakému výsledku dospejeme aj prostredníctvom „univerzálnej" metódy webovej služby **DotazNW**.

```
- <NewDataSet xmlns="">
- <Northwind diffgr:id="Northwind1" msdata:rowOrder="0">
  <nazov>Banany</nazov>
  <cena>39.90</cena>
  </Northwind>
- <Northwind diffgr:id="Northwind2" msdata:rowOrder="1">
  <nazov>Mrkva</nazov>
  <cena>13.30</cena>
  </Northwind>
</NewDataSet>
```

Ak prostredníctvom testovacieho okna webovej služby zadáme SQL príkaz

```
delete from tovar;
```

následným dotazom v konzolovej aplikácii Query Analyzer zistíme, že databázová tabuľka je prázdna.

```
select * from tovar;

nazov                cena
(0 row(s) affected)
```

Takže trocha ironicky môžeme konštatovať, že programové konštrukcie tohoto typu môžeme tvoriť len vtedy ak nás už údaje v databázach omrzeli.

Vytvorenie webovej služby Pôžičková kalkulačka (Visual Basic)

Túto službu vytvoríme hlavne z toho dôvodu, že webovú službu MortgageCalc využíva Microsoft vo svojich cvičných príkladoch a Hands – on laboch. Po zadaní výšky pôžičky, počtu mesiacov a úrokovej miery vypočíta výšku mesačnej splátky

Pri vytváraní webovej služby v jazyku Visual Basic postupujeme podobne ako v predchádzajúcom prípade. Pomocou menu **File | New | Project** vytvoríme v zložke **Visual Basic Project** nový projekt typu **ASP.NET Web Service**. Nazveme ho **wsSplatky**.

Pre výpočet splátky použijeme funkciu **`Pmt`**. Kód funkcie **`CalculateMonthlyPayment`** bude:

```
<WebMethod()> Public Function CalculateMonthlyPayment (ByVal Vyska_pozicky As Double, ByVal
Pocet_Mesiacov As Integer, ByVal Urokova_miera As Double) As String
        Dim dblSplatka As Double
        dblSplatka = -Pmt((Urokova_miera / 1200), Pocet_Mesiacov, Vyska_pozicky)
        Return FormatCurrency(dblSplatka)
  End Function
```

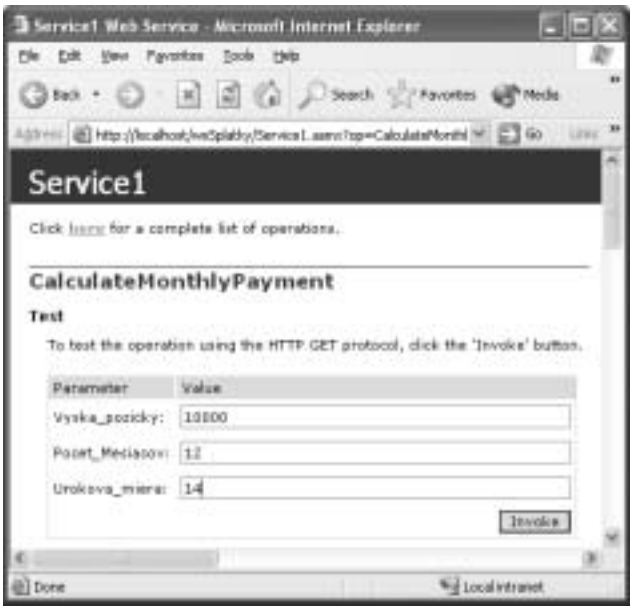
Aj v tomto prípade nahraďme riadok

```
<WebService(Namespace:="http://tmpuri.org/")> _
Public Class Service1
```

riadkom napríklad

```
<WebService(Namespace:="http://lubolacko.sk/brozury/smartdevices/v1")> _
Public Class Service1
```

Pomocou menu **Build | Build wsSplatky** aplikáciu preložíme a zostavíme. Pomocou menu **Debug | start** môžeme webovú službu vyskúšať.



Obrázok 1.13 – Testovanie webovej služby wsSplatky

Po zatlačení tlačidla **Invoke** sa zobrazia údaje poskytnuté webovou službou, v tomto prípade výška mesačnej splátky vo formáte XML.

```
<?xml version="1.0" encoding="utf-8" ?>
  <string xmlns="http://lubolacko.sk/brozury/smartdevices/v1/">897,87 Sk</string>
```

Kapitola 2

eMbedded Visual Tools 3.0, eMbedded Visual C++ 4.0

Vývojové nástroje a utility balíka **Microsoft® eMbedded Visual Tools 3.0** (ďalej len eMVT) vytvárajú kompletné vývojové prostredie pre vývoj aplikácií a softvérových komponentov pre platformy, ktoré sú označované ako Windows Powered, napríklad pre prístroje triedy Pocket PC. Vývojové prostredie **Microsoft® eMbedded Visual C++ 4.0** je následníkom **eMbedded Visual Tools 3.0**, pričom prináša možnosť vývoja aplikácii pre platformu Windows CE .NET.

Obrázok 2.1 – logo eMVT

Veľkou výhodou je autonómnosť eMbedded Visual Tools 3.0, to znamená, že nepotrebujeme mať nainštalované žiadne iné vývojové prostredie, napríklad Visual Studio alebo najnovšiu verziu Visual Studio .NET. Súčasťou inštalácie eMVT sú podporné moduly **Software Development Kit** (SDK) pre ten typ zariadenia, pre ktorý aplikáciu vyvíjame. K dispozícii sú moduly pre Pocket PC a Handheld PC. Je možné doinštalovať moduly pre platformu Smartphone 2002 a Pocket PC 2002. Tieto moduly sú dodávané vo forme SDK (Software Development Kit) . Moduly **Pocket PC 2002 SDK** a **Smartphone 2002 SDK Beta** je možné stiahnuť z webu, alebo objednať na CD.

Vývojové prostredie **eMbedded Visual Tools 3.0** môžeme rozdeliť na dva takmer nezávislé bloky:
- **eMbedded Visual C++® 3.0,**
- **eMbedded Visual Basic® 3.0.**

eMbedded Visual C++® 3.0 (eVC)

Vývojové prostredie obsahuje kompilátor objektovo orientovaného jazyka C++ a linkery natívneho kódu pre procesory používané v mobilných zariadeniach. Vyvíjané aplikácie môžu využívať služby príslušného operačného systému a môžu taktiež cez ovládače využívať vstupne – výstupné zariadenia ako dotykový displej, porty a podobne. Podobne ako pre vývoj aplikácií pre klasické desktop PC, aj v mobilných zariadeniach sa programovací jazyk C++ používa najmä pre aplikácie, kde je primárne vyžadovaný vysoký výkon a samozrejme pre vývoj ovládačov. Ako u „klasických“ Windows aj na mobilným platformách môžeme využívať a vyvíjať komponenty typu COM (Component Object Module) a ActiveX. eVC umožňuje okrem využitia softvérových emulátorov aj ladenie aplikácii na diaľku prostredníctvom pripojenia cez ActiveSync, alebo prostredníctvom LAN.

eMbedded Visual Basic® 3.0 (eVB)

Hoci je eVB je bližší jazyku Visual Basic Script, než jazyku Visual Basic, ktorý poznáme z Visual Studia, eVB je nástroj pre rýchle vytváranie širokého spektra aplikácií, od najjednoduchších až po náročné podnikové projekty typu klient – server. Možnosť rýchleho vývoja je daná implementovaním prostredia typu RAD (Rapid Application Develop). Dôraz sa kladie na jednoduchosť a maximálnu vizualizáciu návrhu. Na rozdiel od kompilátora C++ sa nevytvára priamo natívny kód pre príslušný procesor, ale zdrojový kód v jazyku Visual Basic sa prekladá do takzvaného P-kódu, ktorý je potom interpretovaný. Výhodou je hlavne rýchlosť vývoja aplikácie, preto je Visual Basic vhodný hlavne pre jednoduchšie utility a aplikácie, ktoré je potrebné nasadiť operatívne a pre vývoj prototypov aplikácií.

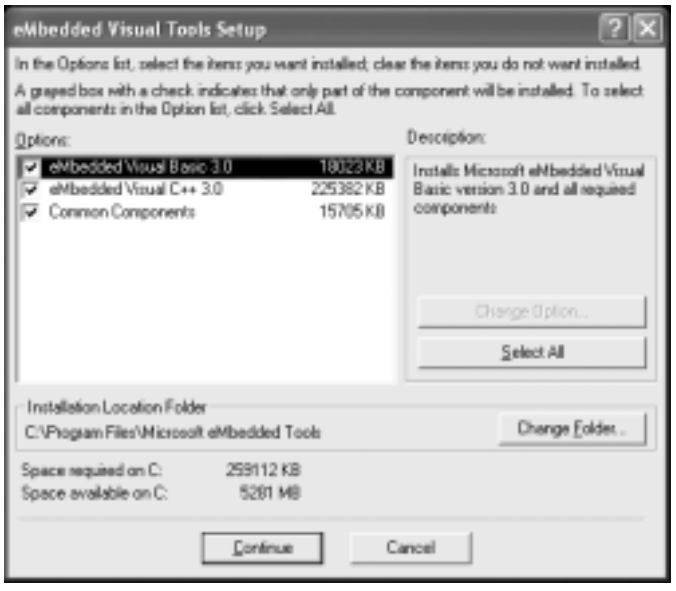
Inštalácia eMbedded Visual Tools 3.0

Minimálne nároky na hardvér sú na dnešnú dobu pomerne skromné:
Procesor: Pentium 150MHz alebo vyšší,
RAM: 24 Mb RAM pre Windows 98 SE, 32 Mb pre Windows NT/ 2000/XP, odporúčané 48 Mb.
Operačný systém: Microsoft Windows 98 SE alebo Microsoft Windows NT +SP5/ 2000/XP
Hard-disk: minimálna inštalácia – 360 Mb; kompletná inštalácia – 720 Mb

Inštalácia produktu je veľmi jednoduchá, pozostáva z niekoľkých krokov. V prvom kroku sa musíme rozhodnúť, pre aké platformy budeme svoje aplikácie vyvíjať. Aplikácia sa štandardne nainštaluje do adresára *C:\Windows CE Tools*.

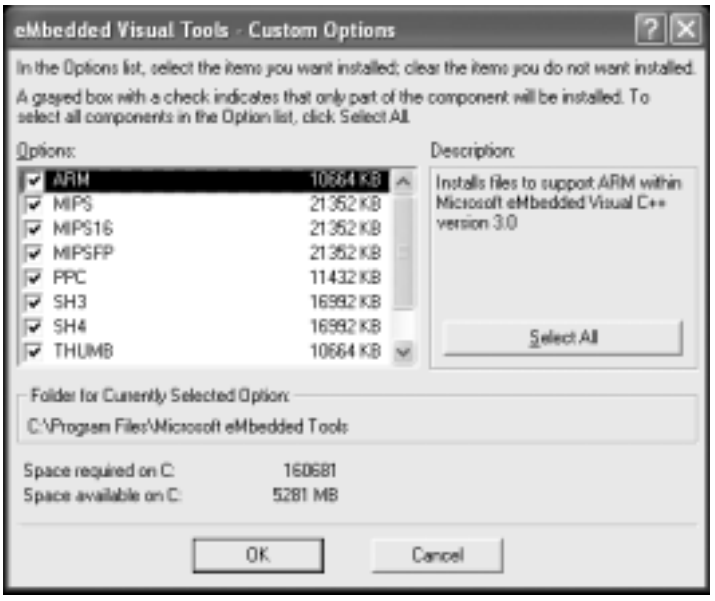
Obrázok 2.2 – Inštalácia: výber produktov

Ak chceme vyvíjať aplikácie len pre platformu Pocket PC 2002, možnosti *Platform SDK (H/PC Pro)* a *Platform SDK (Palm Size PC 1.2)* môžeme nechať nezaškrtnuté.
V nasledujúcom kroku volíme programovací jazyk (Visual Basic, Visual C++) a taktiež, či chceme alebo nechceme nainštalovať podporné komponenty (odporúčame!!).



Obrázok 2.3 – Inštalácia: výber programovacích jazykov

Ak inštalujeme Visual C++, všimnime si tlačidlo **Change Option**. V pomocnom dialógu môžeme vybrať typy procesorov, pre ktoré budeme svoje aplikácie vyvíjať. Pozor, ak chceme svoje aplikácie testovať na platforme PC prostredníctvom softvérového emulátora, musíme ponechať zaškrtnuté voľby X86.



Obrázok 2.4 – Inštalácia: výber procesorov

Microsoft Windows Platform SDK for Pocket PC 2002

Pre vývoj aplikácií pre platformu Pocket PC 2002 je potrebné nainštalovať ešte balíček Microsoft Windows Platform SDK for Pocket PC 2002. Podmienkou je nainštalované vývojové prostredie eMbedded Visual Tools 3.0. Nároky na hardvér a operačný systém sú v tomto prípade oveľa prísnejšie:
Procesor: Pentium II 400MHz alebo vyšší,
RAM: 128 Mb RAM , odporúčané 196 Mb.
Operačný systém: Microsoft Windows 2000/XP
Hard-disk: minimálna inštalácia – 360 Mb; kompletná inštalácia – 720 Mb
Aplikácia sa štandardne nainštaluje do adresára *C:\Windows CE Tools\wce300\Pocket PC 2002*.

Vývoj aplikácie typu Hello Word – Visual C++

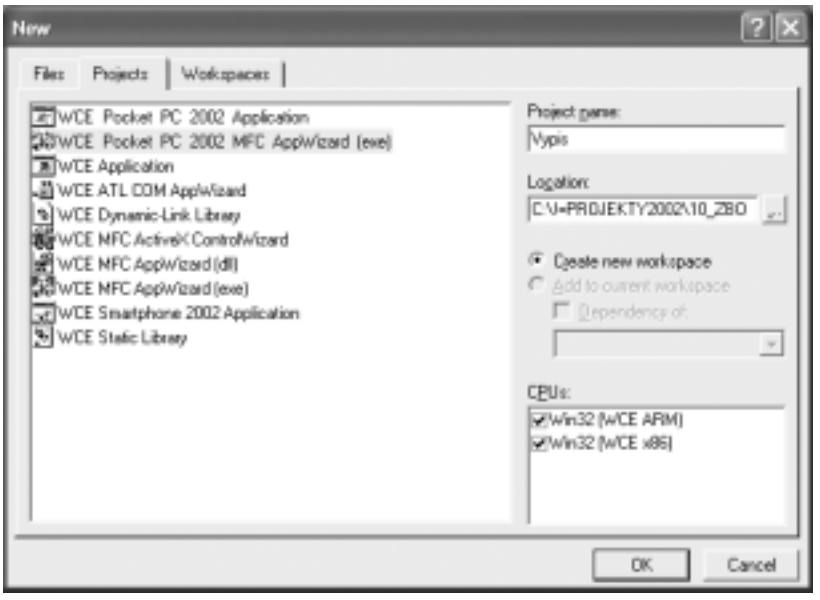
Ako prvú aplikáciu takmer v každej príručke preberajú aplikáciu typu Hello World. Úloha tejto aplikácie je veľmi jednoduchá – vypísať na obrazovku jednoduchý oznam. Nie je to až tak bezvýznamná vec, ako by sa na prvý pohľad zdalo. Niekedy môže byť neznalosť spôsobu výpisu jednoduchého formátovaného textu veľmi limitujúca. Čo máme z toho, ak v C++ bravúrne napíšeme rekurzívny algoritmus pre výpočet čísla PI na 500 desatinných miest, keď potom nedokážeme vypísať výsledok.
My si túto úlohu ešte trochu skomplikujeme, nakoľko budeme komunikovať aj s používateľom, ktorý musí prostredníctvom jednoduchého dialógu zadať svoje meno (reťazec) a vek (číslo). V tomto cvičnom príklade pôjde hlavne o kompletný postup napísania aplikácie a jej odladenie na softvérovom emulátore na PC a následne na jej nainštalovanie a spustenie na príslušnom mobilnom zariadení.

Zadanie úlohy 2.1: Vytvorte aplikáciu, ktorá umožní používateľovi zadať meno a vek a následne tieto údaje vypíše na obrazovke.

Riešenie úlohy 2.1: V tejto prvej cvičnej aplikácii s výhodou využijeme možnosti poskytované objektovou knižnicou MFC (Microsoft Foundation Class). Riešenie rozdelíme do niekoľkých logických krokov. Tí, ktorí s touto knižnicou už majú skúsenosti, napríklad z vývojových prostredí Visual C++ verzie 5.0 a 6.0 majú samozrejme výhodu. Naša prvá cvičná aplikácia bude koncipovaná ako určitý kompromis, aby ju pochopili aj začiatočníci a na druhej strane, aby sa pokročilí príliš nenudili. Pravdepodobne najlepší a najefektívnejší zdroj pre naštudovanie MFC je multimediálne výukové CD Mastering for Visual C++.

Krok 1. – Vytvorenie aplikácie typu MFC pre platformu Pocket PC 2002

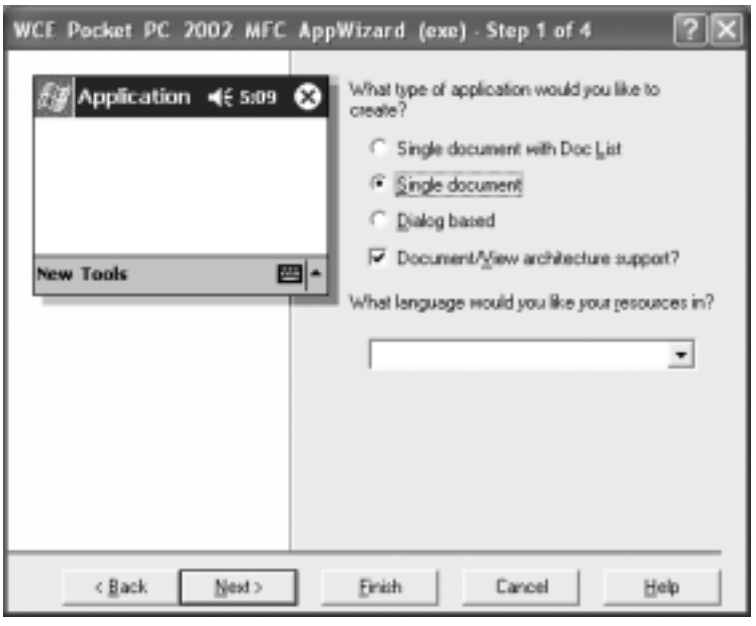
Pomocou menu **File | New** vytvoríme nový projekt typu **WCE Pocket PC 2002 MFC AppWizard (exe)**. V úvodnom dialógu pomenujeme svoj projekt, (v našom prípade Výpis) a špecifikujeme adresár, do ktorého budú súbory tvoriace projekt uložené. Výhodné je projekty umiestňovať do samostatného pracovného adresára, aby sa dali ľahko zálohovať. V pravom dolnom rohu určíme typ procesora, a to vrátane platformy x86, ak chceme tento projekt emulovať na PC.



Obrázok 2.5 – Nový projekt – aplikácia typu MFC

V ďalšom dialógu (Prvý krok aplikačného sprievodcu) vyberieme typ aplikácie. Na výber máme tri možnosti:

- **Single Document with Doc List**
- **Single Document**
- **Dialog based**



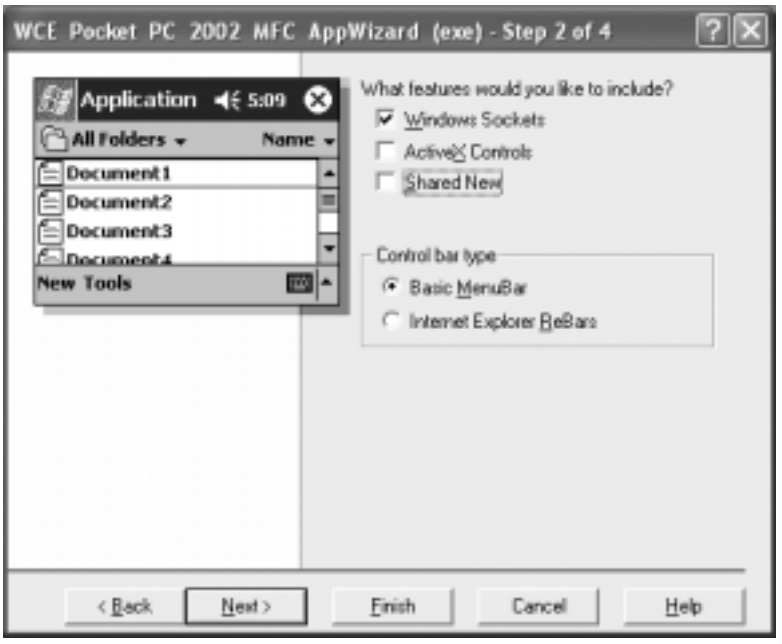
Obrázok 2.6 – výber typu aplikácie

- **Single Document with Doc List** – aplikácia tohoto typu umožňuje pracovať s viacerými dokumentmi rovnakého typu. Napríklad textový editor pracujúci s viacerými dokumentmi typu DOC, TXT a podobne
- **Single Document** projekt tohoto typu je v podstate klasická „Windows“ aplikácia. Ak chceme vyvíjať aplikáciu kde budú rutiny a triedy pracujúce s dokumentom dôsledne oddelené od tried, ktoré zabezpečia zobrazenie dokumentu na obrazovke, môžeme zaškrtnúť políčko ***Document/View Architecture support?***.
- **Dialog based** dominantnou časťou dialógovej aplikácie (napríklad kalkulátor vo Windows) je hlavné dialógové okno pre komunikáciu s používateľom

V našom cvičnom príklade, aby sme splnili zadanie môžeme vytvoriť dialógovú aplikáciu, alebo aplikáciu typu Single Document, ktorá samozrejme taktiež môže využívať dialógy. Pretože je to cvičný príklad, zvolíme z pedagogického

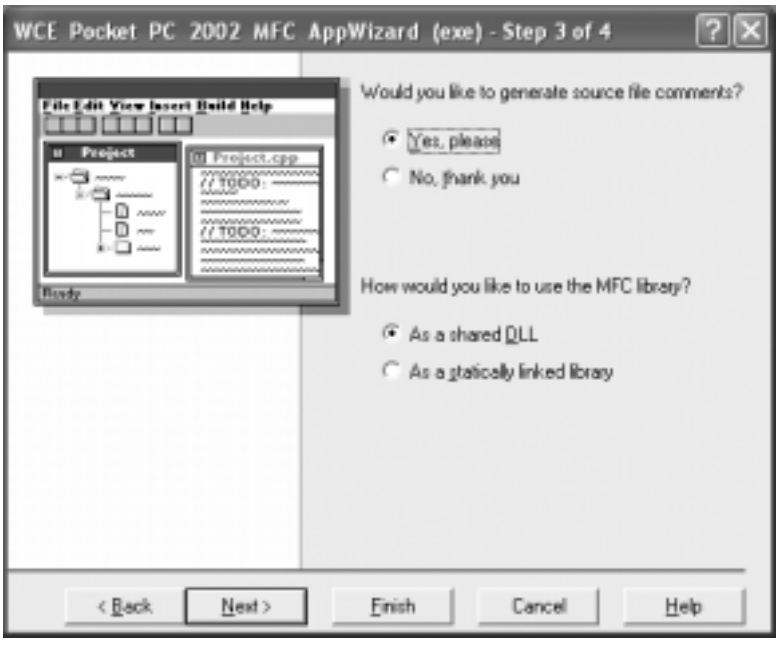
hľadiska pravdepodobne najvýhodnejšie riešenie, teda aplikáciu typu Single Document a zaškrtnime políčko ***Document/View Architecture support?***.

V nasledujúcom dialógu (Druhý krok aplikačného sprievodcu) volíme, aké typy komponentov a technológií mienime vo svojej aplikácii využiť a aký má byť dizajn systému menu našej aplikácie.



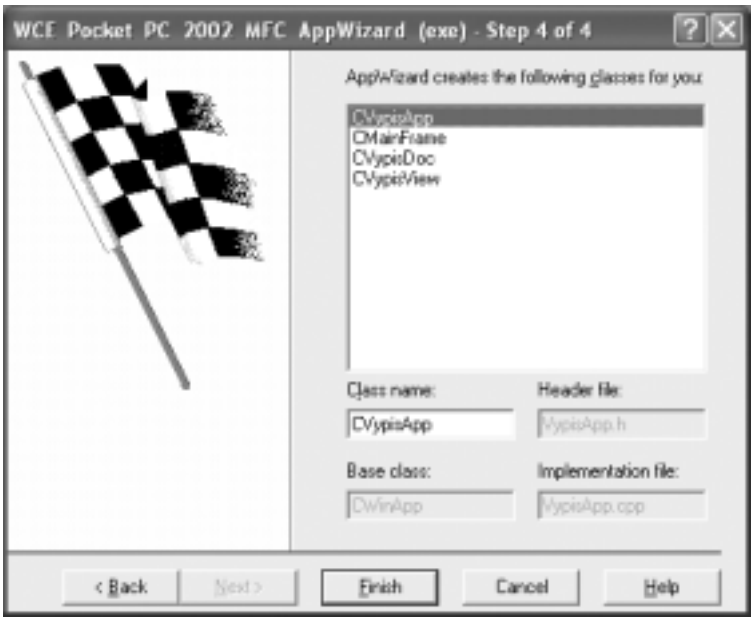
Obrázok 2.7 – Nastavenie parametrov projektu

V nasledujúcom dialógu (Tretí krok aplikačného sprievodcu) špecifikujeme, či má Sprievodca vytvorením aplikácie (Application Wizard) vkladať do nášho projektu komentáre typu To-Do, prostredníctvom ktorých nám naznačí, ktoré časti kódu kam patria. Nie je nijaký dôvod, prečo by sme v našom cvičnom projektu nevyužili takúto pomocnú ruku. V dolnej polovici dialógu špecifikujeme, či sa knižnica MFC nainštaluje ako samostatná DLL, takže ju budú môcť zdieľať všetky aplikácie. Druhá možnosť je prilinkovať túto knižnicu priamo k našej aplikácie. Z hľadiska úspory pamäťovej kapacity je výhodná prvá možnosť.



Obrázok 2.8 – Nastavenie ďalších parametrov projektu

V poslednom dialógu aplikačného sprievodcu (krok 4) môžeme zmeniť názvy súborov pre jednotlivé triedy



Obrázok 2.9 – Finálny dialóg

Pripomeňme, že typická MFC aplikácia s Document/View architektúrou obsahuje triedy **Application**, kde sú základné rutiny pre spustenie a beh aplikácie. Trieda **Frame** zapúzdruje systém radenia a zobrazovania okien. Trieda **Document** zapuzdrí rutiny pre prácu s obsahom dokumentu a trieda **View** zabezpečí zobrazenie jeho obsahu na obrazovke.

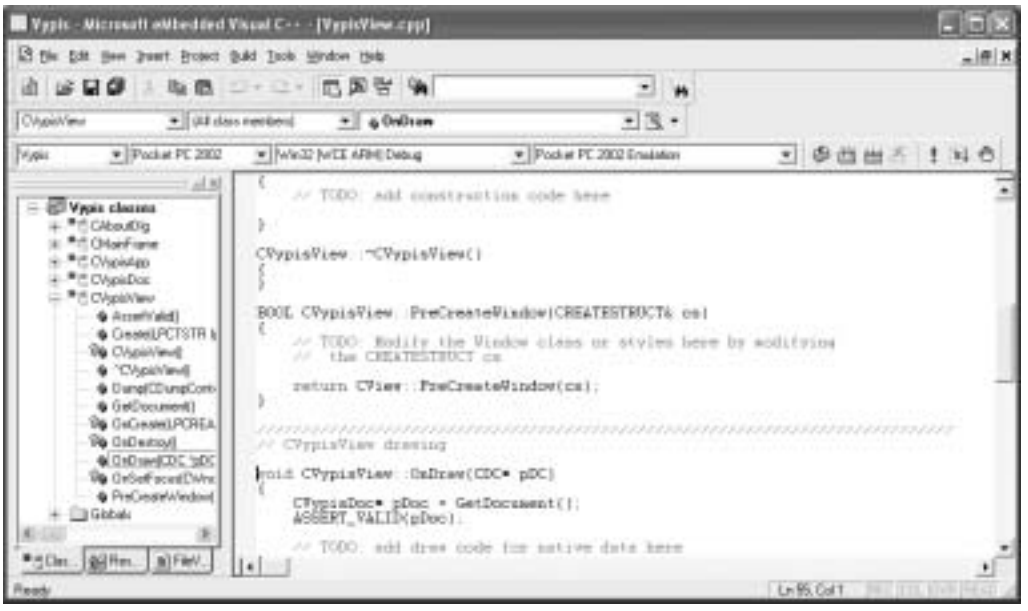
V našom prípade základ aplikácie tvoria triedy:
- **CAboutDlg..** trieda odvodená od **CDialog**, ktorá bude informovať o našej aplikácii,
- **CVypisApp..** trieda odvodená **CWinApp** – hlavný objekt aplikácie,
- **CVypisDoc..** trieda odvodená od **CDocument** – spracovávané údaje,
- **CVypisView..** trieda odvodená od **CView** – pohľad na spracovávané údaje,
- **CMainFrame..** trieda odvodená od **CFrameWnd** – rámové okno aplikácie.

Po zatlačení tlačidla **Finish** sa zobrazia sumárne informácie o práve vytvorenom projekte. V našom prípade:

```
Application type of Vypis:
     Single Document Interface Application, with Doc List support targeting:
          Win32 (WCE ARM)
          Win32 (WCE x86)

Classes to be created:
     Application: CVypisApp in Vypis.h and Vypis.cpp
     Frame: CMainFrame in MainFrm.h and MainFrm.cpp
     Document: CVypisDoc in VypisDoc.h and VypisDoc.cpp
     View: CVypisView in VypisView.h and VypisView.cpp

Features:
    + Menu bar in main frame with menu, buttons, and adornments

    + Windows CE Sockets Support
    + Uses shared DLL implementation
    + Localizable text in:
          English [United States]
```

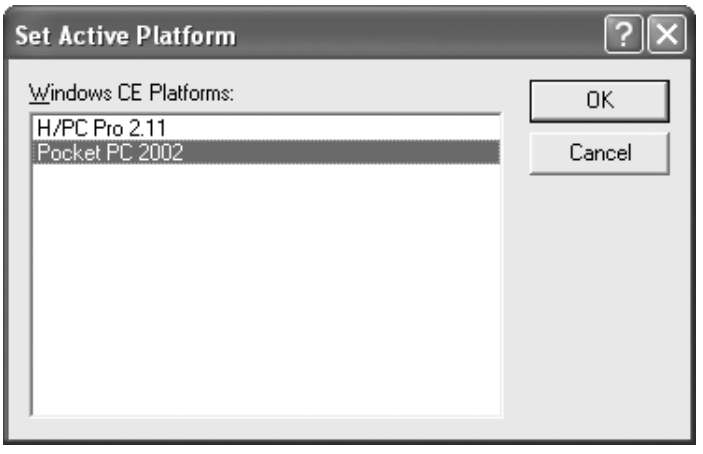
Krok 2. – Preloženie, zostavenie a zavedenie aplikácie do PDA

Dosiaľ sme vlastne bez toho, aby sme napísali čo i len riadok zdrojového kódu, vytvorili jadro aplikácie ktoré zatiaľ nerobí nič. Nič? Až budete ovládať toto vývojové prostredie detailne a skúsili by ste napísať zobrazenie základného okna (zatiaľ prázdneho), systému menu a podobne, ocenili by ste množstvo práce, ktoré vám knižnica MFC a Sprievodca vytvorením aplikácie ušetrí. Pozrime sa na rozloženie pracovného okna vývojového prostredia. Ak poznáme Visual Studio 6.0, alebo Visual Studio.NET, toto okno nám bude dôverne známe.



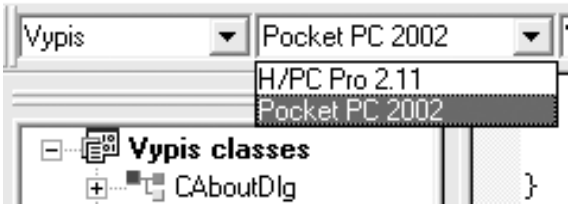
Obrázok 2.10 – Pracovná obrazovka vývojového prostredia

V ľavej časti máme okno, kde môžeme pomocou záložiek zobraziť zoznam použitých tried, resource a zoznam súborov tvoriacich projekt. V pravej časti okna zobrazujeme konkrétnu entitu, napríklad zdrojový kód, návrhové okno dialógu bitmapy a podobne. Našou úlohou v tomto kroku však nie je ani tak spoznávanie vývojového prostredia, ako skôr preloženie a spustenie práve vytvorenej „prázdnej“ aplikácie jednak na PC na softvérovom emulátore a taktiež aj na konkrétnom zariadení. Najskôr skontrolujeme, prípadne nastavíme platformu pre ktorú budeme túto aplikáciu vyvíjať. V hlavnom menu **Build | Set Active Platform** aktivujeme príslušný dialóg a nastavíme platformu **Pocket PC 2002**.



Obrázok 2.11 – Nastavenie aktívnej platformy pomocou dialógu

Aktívnu platformu môžeme nastaviť aj pomocou roletových ovládacích prvkov toolbaru.



Obrázok 2.12 – Nastavenie aktívnej platformy pomocou toolbaru

Po tomto kroku sa pomaly blížime do finále. Cieľom je preložiť aplikáciu a spustiť na emulátore. Podobne ako sme nastavili aktívnu platformu, nastavíme pomocou toolbaru aj to, pre aký procesor sa má aplikácia preložiť a zostaviť. Keďže všetky PC a teda aj naše na ktorom budeme aplikáciu emulovať majú procesor triedy x86, nastavíme voľbu **Win 32 (WCE x86) Debugg**. Neskôr, keď budeme chcieť aplikáciu spustiť v reálnom PDA, vyberieme príslušný procesor, napríklad pre iPAQ to bude procesor ARM a podobne. V hlavnom menu potom aktivujeme voľby **Build | Build Vypis.exe** . V spodnom okne vývojového prostredia potom môžeme sledovať priebeh prekladu, zostavovania a prenosu našej aplikácie do emulátora.

```
------------------Configuration: Vypis - Win32 (WCE x86) Debug--------------------
Compiling resources...
Compiling...
StdAfx.cpp
Compiling...
MainFrm.cpp
Vypis.cpp
VypisDoc.cpp
VypisView.cpp
Generating Code...
Linking...

Vypis.exe - 0 error(s), 0 warning(s)
Downloading files
Checking remote file: \Windows\Start Menu\Vypis.exe.
Downloading file c:\!=projekty2002\10_zbornik pda\...\vypis\x86dbg\vypis.exe.
Checking remote file: \WINDOWS\mfcce300d.dll.
Downloading file c:\windows ce tools\wce300\...\lib\x86\mfcce300d.dll.
Finished downloading.
```

Všetko prebehlo v naprostom poriadku. Tým že sme zatiaľ nepísali žiadny vlastný kód, nemohli sme vlastne s súlade s Murphyho zákonmi ani urobiť žiadne chyby. Všimnime si, že pri nahrávaní aplikácie či už do emulátora, alebo do konkrétneho zariadenia sa kontroluje, či je v mobilnom zariadení knižnica MFC DLL, v našom prípade konkrétne súbor **`mfcce300d.dll`**. Ak je všetko v poriadku, malo by sa aktivovať okno emulátora. Po zatlačení tlačidla Start v emulátore uvidíme ikonu našej aplikácie Výpis v hlavnom menu

Obrázok 2.13 – Vývojové prostredie + emulátor

Samozrejme že neodoláme pokušeniu a našu prázdnu aplikáciu si v emulátore spustíme. Zatiaľ presne podľa očakávania cvičná aplikácia Výpis nič nerobí ani nič neukazuje.

Obrázok 2.14 – Prvé spustenie cvičnej aplikácie

Aby sme mohli v ďalších etapách vývoja nahrať do emulátora túto aplikáciu znovu, musíme ju zakaždým ukončiť. Najjednoduchšie to urobíme tak, že si kliknutím na ikonu v pravom dolnom rohu necháme zobraziť virtuálnu klávesnicu a postupne na nej zatlačíme symboly **CTRL** a **Q**.

Krok 3. – Naprogramovanie aplikačnej logiky

Ďalší postup návrhu nášho projektu v podstate rozdeliť do dvoch etáp: Prvou etapou bude **návrh vizuálnej časti aplikácie** t.j. grafický, logický a funkčný návrh formulárov pre komunikáciu s používateľom. V druhej etape navrhneme **funkčnú logiku aplikácie**. Programátori, ktorí programujú aplikácie pre operačný systém Windows vedia, že väčšina aplikácií pre túto platformu je riadená udalosťami. Niekto pohne myšou, stlačí klávesu príde správa po sériovom alebo USB porte a podobne. Preto v tejto etape hlavne programujeme obsluhu udalostí – t.j kód, ktorý sa aktivuje, ak príslušná udalosť nastane. Napríklad reakciou na to, že používateľ presunie ukazovateľ myši pri zatlačenom tlačidle na iné miesto musí byť presunutie objektu, napríklad grafického symbolu, na ktorý myš predtým ukazovala. Ako prvý krok (aby sme dodržali tradíciu Hello World), bude jednoduchý výpis textu na obrazovku. Výpis obsahu dokumentu na obrazovku a teda aj akýkoľvek výpis, nakoľko v našej jednoduchej cvičnej aplikácii so žiadnym reálnym dokumentom nepracujeme má na starosti trieda CView, v našom prípade konkrétne trieda **`CVypisView`**. Kód pre výpis textu preto umiestnime do metódy **`OnDraw`**, ktorá sa volá vždy pri vykresľovaní obsahu. Náš kód pre výpis textu (vo výpise vyznačený hrubým fontom) dopíšeme presne tam kam patrí, teda za komentár `// TODO: add draw code for native data here:`

```
void CVypisView::OnDraw(CDC* pDC)
{
        CVypisDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
        // TODO: add draw code for native data here
       CRect rect;
        GetClientRect(rect);
        pDC->DrawText("Cvicna aplikacia VYPIS", rect, DT_CENTER | DT_VCENTER);
}
```

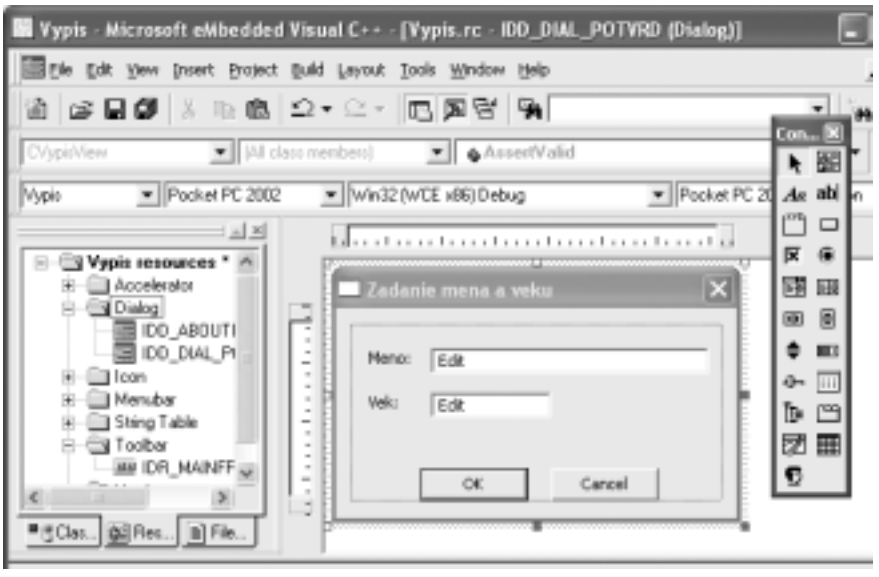
Po preložení a spustení aplikácie vidíme, že veta „...aplikácia, ktorá nerobí nič...“ prestáva platiť. Naša aplikácia vypísala na obrazovku textový reťazec.

Obrázok 2.15 – Výpis jednoduchého textového reťazca

Aby sme splnili literu zadania, potrebujeme získať od používateľa jeho meno a vek a tieto údaje potom vypísať na displej. Preto pre tento účel vytvoríme jednoduchý zadávací dialóg. Zadávací dialóg (vo všeobecnosti akúkoľvek funkciu a službu, ktorá sa vykoná na pokyn používateľa) môžeme aktivovať pomocou menu, alebo kliknutím na tlačidlo toolbaru. Pri pozornejšom pohľade na okno našej aplikácie vidíme, že Sprievodca vytvorením aplikácie už vygeneroval v spodnej časti jednoduché menu (obsahuje položky **Edit** a **Tools**) a jednoduchý toolbar s dosiaľ neaktívnymi ikonami nožníc, kopírovania a aktovky. Po aktivovaní záložky **Resources** v ľavej časti okna sa zobrazí stromová štruktúra jednotlivých skupín prvkov. Nás budú zaujímať záložky **Dialog**, **Toolbar** a **Menubar**.

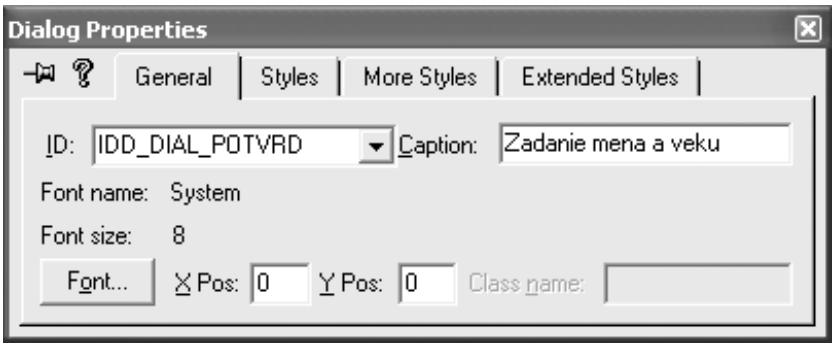
Dialog

Najskôr navrhneme zadávací dialóg. Kliknutím pravého tlačidla myši na zložku **Dialóg** sa sprístupní kontextové menu. V tomto menu zvolíme položku **Insert dialog**. V novovytvorenom dialógovom okne popremiestňujeme tlačidlá **OK** a **Cancel** do spodnej časti dialógového okna. Ak nám záleží na lokalizácii môžeme ich premenovať na **Potvrď** a **Storno**. Na plochu dialógu umiestnime dva nápisy Meno: a Vek: typu **Static Text** a dva zadávacie okná typu **Edit Box**. Tieto pomenujeme napríklad IDC_EDIT_MENO a IDC_EDIT_VEK. Vzhľad nášho dialógového okna by mohol byť napríklad takýto.



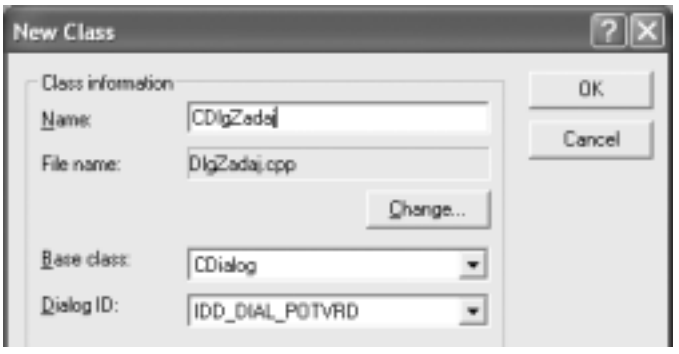
Obrázok 2.16 – Návrh zadávacieho dialógu

Návrh vzhľadu dialógu ukončíme jeho pomenovaním a doplnením názvu jeho záhlavia



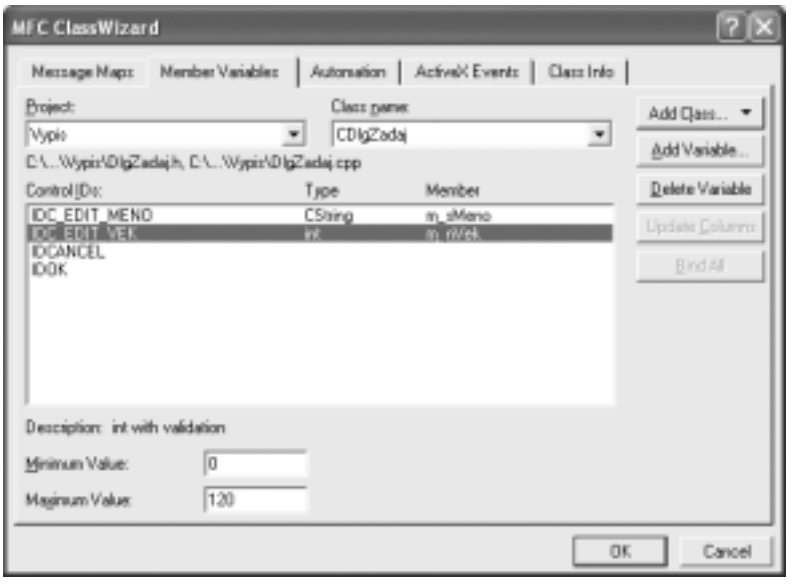
Obrázok 2.17 – Parametre zadávacieho dialógu

Po týchto krokoch máme návrhovú etapu dialógu za sebou. Aby sa dialóg nielen zobrazil, ale aj fungoval, to znamená aby odovzdal aplikácii zadané údaje, musíme vytvoriť k novo navrhnutému dialógu triedu. Použijeme sprievodcu pre vytvorenie triedy Class Wizard. Aktivujeme ho buď pomocou menu **View | Class Wizard**, alebo klávesovou skratkou CTRL-W. Potvrdíme voľbu **Create new class**.



Obrázok 2.18 – Vytvorenie triedy k zadávacieho dialógu

Trieda **CDlgZadaj** bude odvodená od triedy CDialog a v našom prípade bude okrem zdedených objektov zapúzdrovať len dve členské premenné **m_sMeno** (typu CString) a **m_nVek** (typu Integer). Tieto premenné navrhneme taktiež pomocou Class Wizarda v záložke **Member Variables**, tlačidlom **Add Class**. Všimnime si aj validačné možnosti v dolnej časti návrhového dialógu. U reťazcovej premennej môžeme limitovať maximálnu dĺžku a u číselnej premennej jej rozsah.



Obrázok 2.19 – Návrh členských premenných triedy

Týmto krokom je návrh zadávacieho dialógu u konca. Ak si pozrieme deklaráciu triedy (súbor DlgZadaj.h), Class Wizard vygeneroval kód:

```
class CDlgZadaj : public CDialog
{
// Construction
public:
    CDlgZadaj(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
    //{{AFX_DATA(CDlgZadaj)
    enum { IDD = IDD_DIAL_POTVRD };
    CString    m_sMeno;
    int        m_nVek;
    //}}AFX_DATA
...
```

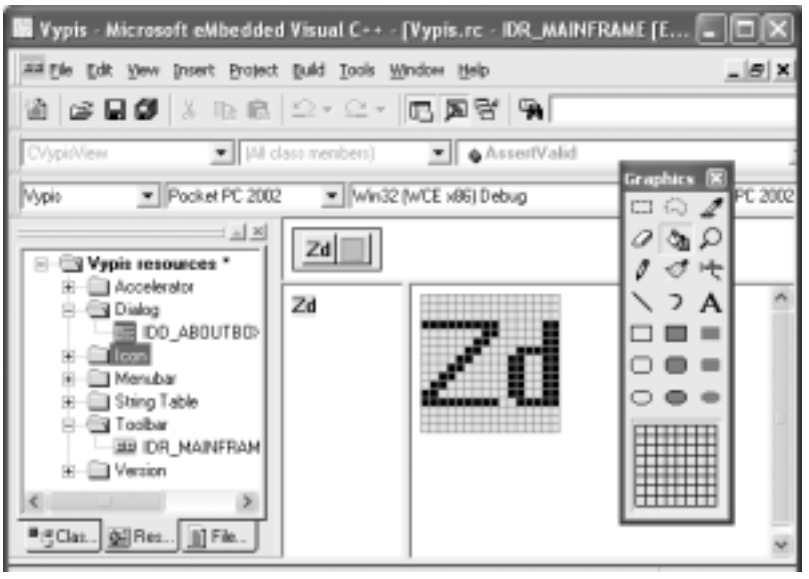
Ukladanie údajov z edit boxov do členských premenných je zaistený kódom (súbor DlgZadaj.cpp), ktorý taktiež vygeneroval Class Wizard:

```
void CDlgZadaj::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CDlgZadaj)
    DDX_Text(pDX, IDC_EDIT_MENO, m_sMeno);
    DDV_MaxChars(pDX, m_sMeno, 50);
    DDX_Text(pDX, IDC_EDIT_VEK, m_nVek);
    DDV_MinMaxInt(pDX, m_nVek, 0, 120);
    //}}AFX_DATA_MAP
}
```

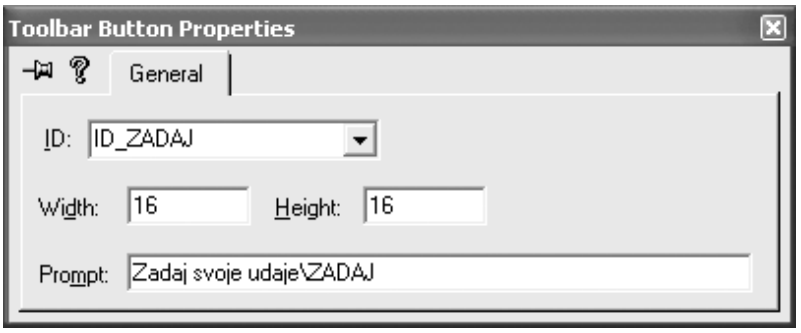
Zostáva nám nadviazať dialóg, respektíve triedu CDlgZadaj na zvyšok aplikácie. V tejto cvičnej aplikácii ukážeme aktiváciu zadávacieho dialógu, pomocou toolbaru.

Toolbar

Po označení entity IDR_MAINFRAME môžeme editovať tlačidlovú lištu ako celok, napríklad metódou drag and drop „odstrániť" nepotrebné tlačidlá z tlačidlovej lišty. V našom projekte môžeme pokojne odstrániť všetky tlačidlá a vytvoriť jedno nové napríklad s grafickým symbolom ZD (zadávací dialóg).

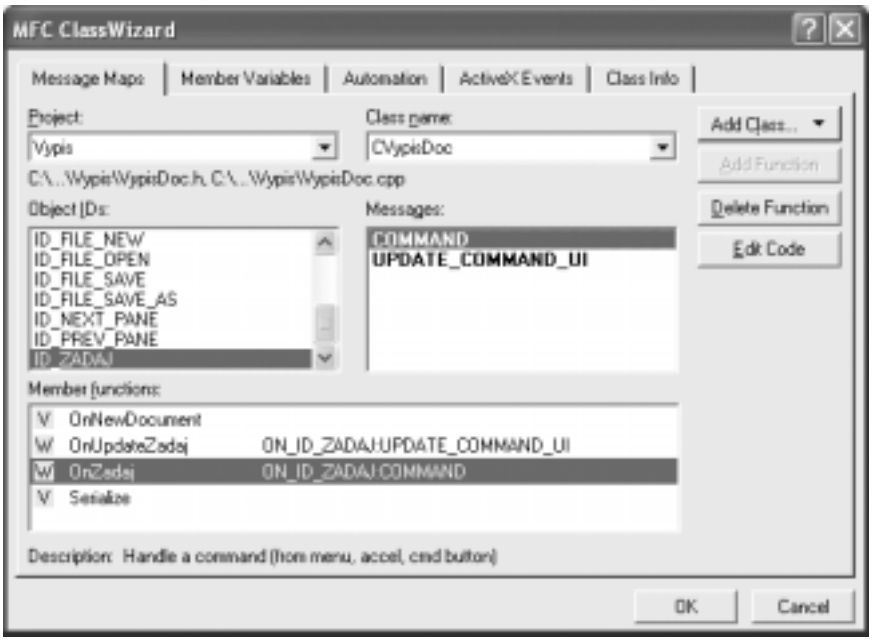


Obrázok 2.20 – Vytvorenie tlačidiel toolbaru



Obrázok 2.21 – Parametre tlačidla toolbaru

Pomocou Class Wizardu (tlačidlo **Add Function**) vytvoríme procedúru **OnZadaj**, prípadne aj **OnUpdateZadaj** ak máme v úmysle meniť vlastnosti tlačidla toolbaru, napríklad toto tlačidlo za určitých okolností znefunkčniť a podobne. Procedúry sú vytvorené ako členské v triede CVypisDoc.

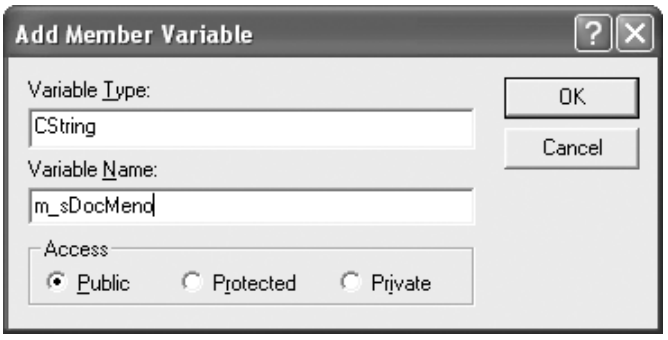


Obrázok 2.22 – Vytvorenie obslužných procedúr v triede CVypisDoc

Class Wizard vygeneruje prázdne telo procedúry. Zostáva už len doprogramovať aktivovanie dialógu a nastavenie hodnôt systémových premenných.

```
void CVypisDoc::OnZadaj()
{
    // TODO: Add your command handler code here
}
```

V triede CVypisDoc vytvoríme dve členské premenné **m_sDocMeno** (typu CString) a **m_nDocVek** (typu Integer). Členskú premennú vytvoríme najjednoduchšie pomocou kontextového menu, ktoré sa zobrazí ak klikneme ľavým tlačidlom myši na názov triede v ľavom okne v záložke **Class**. Použijeme položku menu **Add member variable**.... a v dialógovom okne vyplníme typ premennej a jej názov. Premenné budú typu Public, nakoľko k nim budeme neskôr pristupovať z triedy CVypisView.



Obrázok 2.23 – Vytvorenie členských premenných v triede CVypisDoc

V konštruktore triedy CVypisDoc nastavíme počiatočnú hodnoty premennej **`m_nDocVek`** na hodnotu NULL.

```
CVypisDoc::CVypisDoc()
{
 // TODO: add one-time construction code here
     m_nDocVek = NULL;
}
```

Teraz pridáme odkaz na hlavičkový súbor obslužnej triedy dialógu do súboru VypisDoc

```
// VypisDoc.h : interface of the CVypisDoc class
/////////////////////////////////////////////////////////////////////////////

#include "DlgZadaj.h"
...
```

a v procedúre OnZadaj aktivujeme dialóg ako modálny. Kompletný kód procedúry OnZadaj potom bude:

```
void CVypisDoc::OnZadaj()
{
    // TODO: Add your command handler code here
    CDlgZadaj dZD;
    if(dZD.DoModal()==IDOK)
    {
         m_sDocMeno = dZD.m_sMeno;
         m_nDocVek = dZD.m_nVek;

    }
    UpdateAllViews(NULL);
}
```

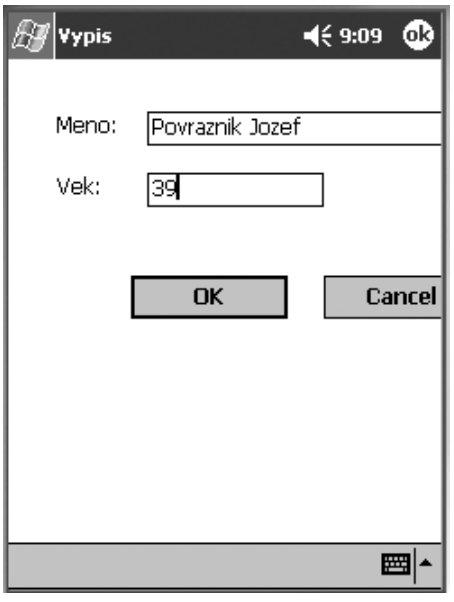
Ak v dialógu stlačíme tlačidlo OK, a teda podmienka **`if(dZD.DoModal()==IDOK)`** bude splnená, prenesú sa zadané údaje do členských premenných dokumentu. Procedúra **`UpdateAllViews(NULL);`** aktivuje obnovenie zobrazenia obsahu displeja, inými slovami v triede CVypisView sa aktivuje procedúra OnDraw. Doteraz máme v tejto procedúre len výpis jednoduchého textového reťazca:

```
void CVypisView::OnDraw(CDC* pDC)
{
    CVypisDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    // TODO: add draw code for native data here
         CRect rect;
    GetClientRect(rect);
    pDC->DrawText("Cvicna aplikacia VYPIS", rect, DT_CENTER | DT_VCENTER);
}
```

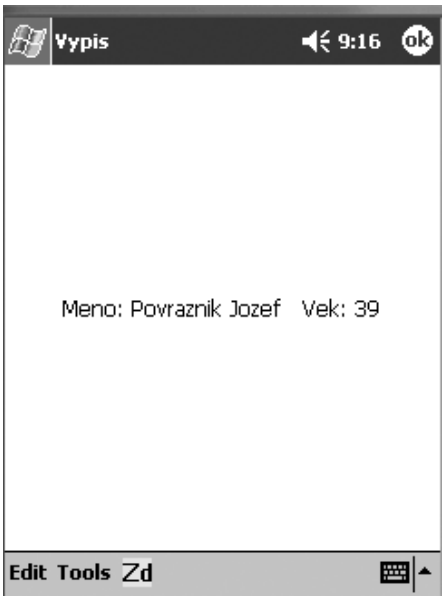
Aby jej funkčnosť zostala zachovaná, úvodný text vypíšeme len v prípade, ak obsah premennej **m_nDocVek** bude mať hodnotu NULL, teda pri spustení aplikácie. V opačnom prípade vypíšeme iný reťazec, ktorý bude obsahovať zadané hodnoty. Všimnime si, že v procedúre už máme získaný pointer **pDoc** na triedu CVypisDoc a teda k jej premenným typu PUBLIC sa ľahko dostaneme. Upravená procedúra OnDraw potom bude

```
void CVypisView::OnDraw(CDC* pDC)
{
    CVypisDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    // TODO: add draw code for native data here
        CString sText;
    CRect rect;
    GetClientRect(rect);

    if(pDoc->m_nDocVek==NULL)
      sText = "Cvicna aplikacia VYPIS";
    else
      sText.Format(_T("Meno: %s    Vek: %d"), pDoc->m_sDocMeno, pDoc->m_nDocVek);
    pDC->DrawText(sText, rect, DT_CENTER | DT_VCENTER);
}
```

Výsledkom podľa emulátora je funkčná aplikácia:



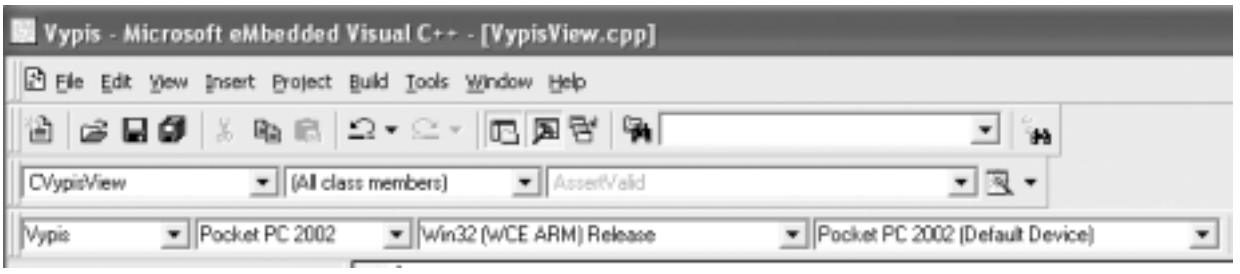
Obrázok 2.24 – Beh aplikácie – zadávací dialóg



Obrázok 2.25 – Beh aplikácie – výpis výsledkov

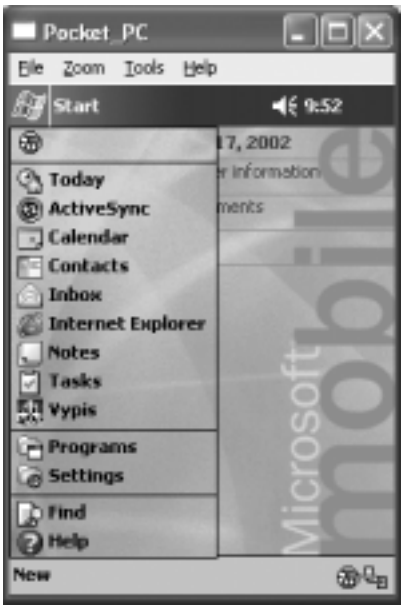
Krok 4. – Prenos aplikácie do reálneho zariadenia

Aplikáciu máme vyvinutú a odladenú na emulátore, je teda najvyšší čas ju preniesť na mobilné zariadenie a vyskúšať ju v reálnom zariadení. Niektoré aplikácie, ktoré pracujú s portami, alebo prídavnými kartami, ktoré sa nedajú simulovať, napríklad GPS, musíme ladiť na reálnom zariadení úplne od začiatku. V takomto prípade nám emulátor pochopiteľne nepomôže. Musíme mať už nainštalovaný a spustený synchronizačný program Microsoft ActiveSync aby bolo možné preniesť našu vyvinutú aplikáciu do reálneho zariadenia. Nastavíme teda pomocou toolbaru reálny procesor, v našom prípade ARM. My sme nastavili verziu Release, nakoľko aplikáciu už máme odladenú na emulátore. Namiesto emulátora nastavíme Default Device.

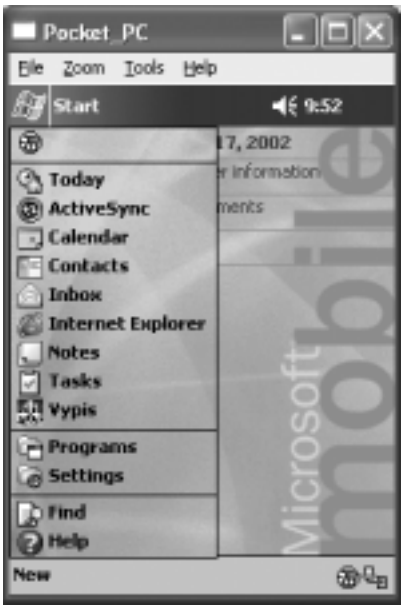


Obrázok 2.26 – Nastavenie vývojového prostredia pre konkrétny procesor a reálne zariadenie

Po preklade a prenose súborov môžeme spustiť našu aplikáciu prostredníctvom menu na reálnom zariadení.



Obrázok 2.27 – Menu reálneho zariadenia



Obrázok 2.28 – Beh aplikácie na reálnom zariadení

Vývoj aplikácie typu Scribble

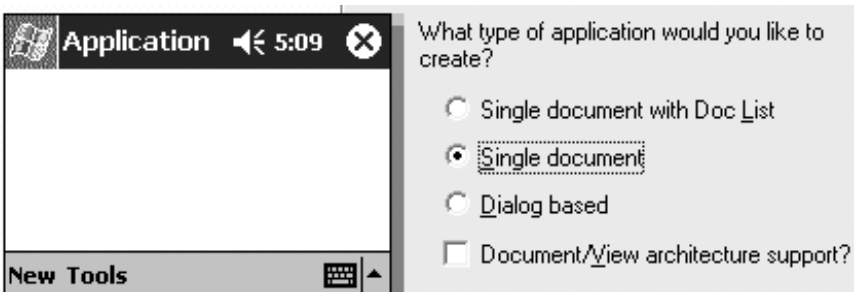
Ako námet pre druhú aplikáciu použijeme taktiež „klasiku“, známy tutoriál pre vývojárov začínajúcich s Visual Studiom, aplikáciu typu Scribble, ktorá nám umožní kresliť perom po obrazovke. Pokročilejší si môžu túto aplikáciu vylepšiť pridaním .NET webovej služby a kódu pre kreslenie na displej iného zariadenia a podobne. Pretože postup vytvorenia aplikácie poznáme z predchádzajúceho príkladu, popíšeme vytvorenie „prázdnej“ aplikácie len heslovite. Táto aplikácia vyžaduje vytvorenie nového projektu a je úplne nezávislá od predchádzajúceho cvičného príkladu.

Zadanie úlohy číslo 2.2: Vytvorte aplikáciu, ktorá umožní používateľovi kresliť čiary na obrazovke.

Riešenie úlohy číslo 2.2: Aj v tejto aplikácii s výhodou využijeme možnosti poskytované objektovou knižnicou **MFC** (Microsoft Foundation Class). Pre uloženie jednotlivých vektorov, z ktorých sa budú skladať nakreslené čiary môžeme využiť proces zvaný serializácia.

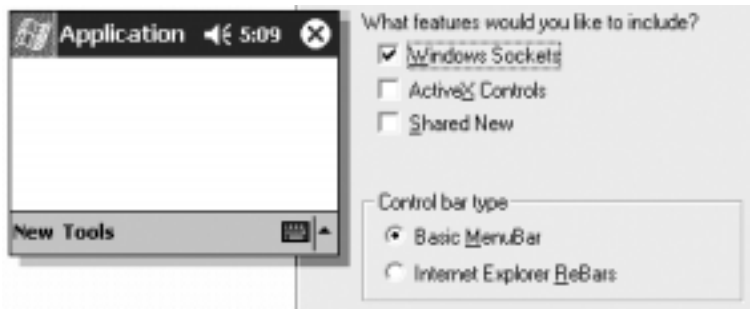
Krok 1. – Vytvorenie aplikácie typu MFC bez architektúry Document / View

Pomocou menu **File | New** vytvoríme nový projekt typu **WCE Pocket PC 2002 MFC AppWizard (exe)**. Projekt môžeme nazvať napríklad **SuperScribble**. Projekt bude typu **Single Document**, pričom políčko pri možnosti **Document / View Architecture** necháme nezaškrtnuté.



Obrázok 2.29 – typ projektu

V ďalšom dialógu (pre istotu ak by sme neskôr skúšali komunikáciu s inými zariadeniami) zaškrtneme možnosť Windows Sockets, pričom políčko Shared New necháme nezaškrtnuté.



Obrázok 2.30 – typ projektu

V nasledujúcom dialógu samozrejme povolíme komentáre typu To-Do, a nastavíme zdieľanie knižníc DLL. Tým je návrh kostry aplikácie ukončený. Aplikáciu môžeme zostaviť a vyskúšať na emulátore.

Krok 2. – Naprogramovanie aplikačnej logiky – rutiny pre kreslenie

Cieľom tohoto kroku je vytvoriť aplikačnú logiku, ktorá umožní kreslenie čiary perom po dotykovom displeji.

– vo vnútri triedy **CChildView** vytvoríme členskú premennú typu CPoint s názvom **m_CurentPoint**. Premenná bude typu Protected.

Obrázok 2.31 – Vytvorenie premennej m_CurrentPoint

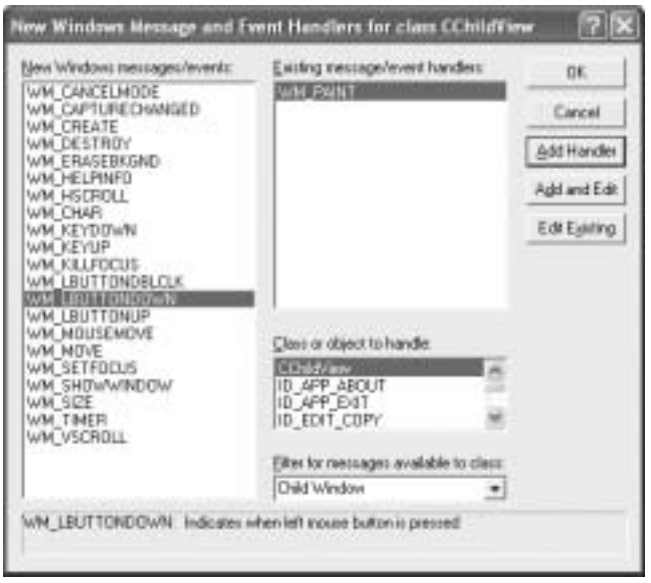
Rovnako vytvoríme aj členskú premennú typu **bool** s názvom **m_InDraw**. Aj táto premenná bude typu Protected.

– k definícii konštruktora triedy **CChildView** (v súbore ChildView.cpp) pridáme kód pre inicializáciu týchto premenných **`:m_InDraw(false),m_CurrentPoint(-1,-1)`**. Kód konštruktora bude potom:

```
CChildView::CChildView() :m_InDraw(false) ,m_CurrentPoint(-1,-1)
{
}
```

– V kontextovom menu zobrazenom po kliknutí ľavým tlačidlom myši na názov triedy **CChildView** vyberieme možnosť **Add Windows Message Handler**. V dialógu v rámiku **Class or object to handle** označíme položku **CChildView**.

V rámiku **Filter for messages available** označíme položku **Child Window**. V ľavom zozname dialógu označíme udalosť **WM_LBUTTONDOWN** (zatlačenie ľavého tlačidla myši, v tomto prípade položenie pera). Tlačidlom **Add and Edit** pridáme obslužnú procedúru pre túto udalosť.



Obrázok 2.32 – Vytvorenie obslužnej procedúry pre udalosť WM_LBUTTONDOWN

Wizard vygeneroval prázdnu obslužnú procedúru:

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    CWnd ::OnLButtonDown(nFlags, point);
}
```

Do procedúry doplníme kód

```
     m_InDraw = true;
     m_CurrentPoint = point;
```

a odstránime riadok `CWnd ::OnLButtonDown(nFlags, point);` Kód procedúry potom bude:

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
  // TODO: Add your message handler code here and/or call default
      m_InDraw = true;
      m_CurrentPoint = point;
}
```

Podobne pridáme obslužnú procedúru pre udalosť **WM_LBUTTONUP**. Kód tejto obslužnej procedúry bude:

```
void CChildView::OnLButtonUp(UINT nFlags, CPoint point)
{
  // TODO: Add your message handler code here and/or call default
    m_InDraw = false;
    m_CurrentPoint = CPoint(-1,-1);
}
```

Položenie a zdvihnutie pera máme ošetrené, zostáva ošetriť jeho pohyb po dotykovom displeji, teda udalosť **WM_MOUSEMOVE**. Kód tejto obslužnej procedúry bude o niečo zložitejší:

```
void CChildView::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    if (m_InDraw)
    {
      if (point != m_CurrentPoint)
      {
        CPoint line[2];
        line[0] = m_CurrentPoint;
        line[1] = point;
        m_CurrentPoint = point;
        CDC* pDC = GetDC();
        pDC->Polyline(line, 2);
        ReleaseDC(pDC);
      }
    }
    else
    CWnd ::OnMouseMove(nFlags, point);
}
```

Takto naprogramovanú aplikáciu môžeme zostaviť a otestovať pomocou emulátora prípadne aj na reálnom mobilnom zariadení.

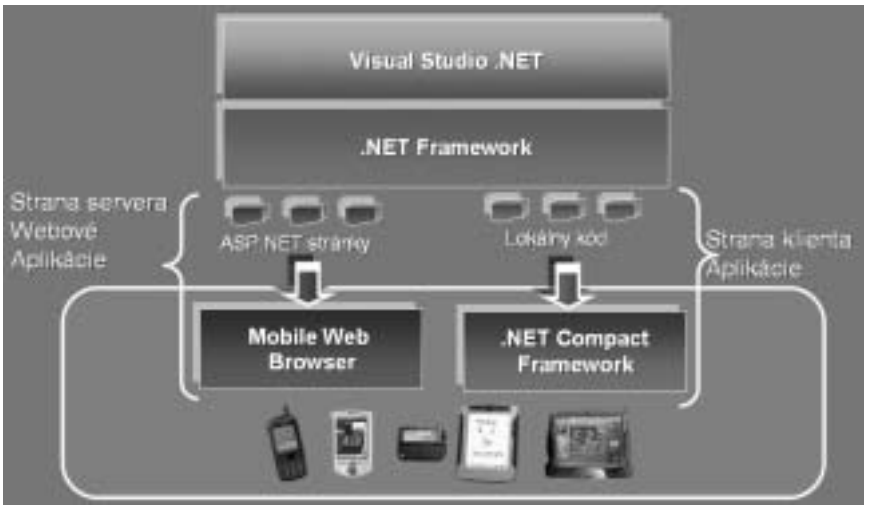
Obrázok 2.33 – Test cvičnej aplikácie SuperScribble

Týmto krokom je naprogramovanie jednoduchej aplikácie typu Scribble ukončené

Kapitola 3

Compact .NET Framework & SDE

Ako bolo uvedené v prvej kapitole, ak chceme vyvíjať projekt pre mobilné zariadenie, máme na výber dve možnosti. Buď ho realizujeme vo forme webovej aplikácie, napríklad pomocou technológie ASP.NET stránok a vtedy použijeme **Microsoft Mobile Internet Toolkit**, alebo ako klasickú aplikáciu prostredníctvom Visual Studia.NET 2003, prípadne Visual Studia.NET rozšíreného o **Smart Device Extensions**.



Obrázok – .NET Compact Framework

Microsoft Smart Device Extensions

Nová verzia **Visual Studio .NET 2003** (kódové označenie Everett) má už modul Smart Device Extensions (ďalej skrátene SDE) priamo integrovaný. Pre staršiu (v dobe písania zborníka aktuálnu produkčnú) verziu Visual Studio.NET je potrebné nainštalovať beta verziu softvérového balíku Microsoft Smart Device Extensions ako nadstavbu vývojového prostredia. Vzhľadom na príchod novej verzie VS.NET 2003 sa uvedenie finálnej verzie SDE kitu ani nepredpokladá. Predmetom tejto kapitoly nie je predstavovať nové vývojové prostredie VS.NET 2003 Everett, ale vývoj aplikácií pre mobilné zariadenia. Všetky príklady a obrázky sú z dôvodu aktuálnosti realizované betaverzii VS.NET 2003, pričom ale boli zároveň otestované aj v staršej verzii. VS.NET+SDE.

Zoznámenie sa s vývojovým prostredím

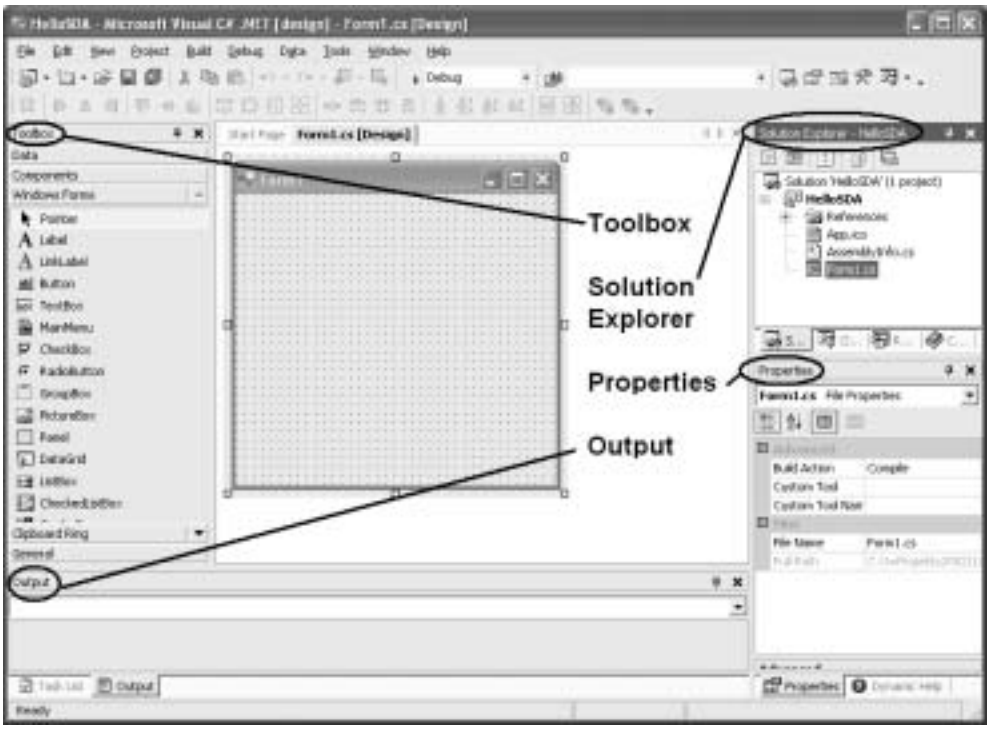
Po spustení vývojového prostredia Visual Studio.NET si všimnime ikony **Smart Device Application** v dialógu **New Project** pre vytvorenie nového projektu v okne **Templates**. Týka sa to zložiek **Visual Basic Projects** a **Visual C# Projects**. **Visual Basic** je svojmu okruhu pomerne známy, takže popisovať ho v tomto zborníku by bolo nosenie dreva do lesa, no priaznivci C siahnu po Visual C#. Tí flexibilnejší s radosťou v očakávaní pozitívnych zmien a tí konzervatívnejší, ktorí na C++ a hlavne jeho pointre nedali dopustiť si budú musieť zvyknúť, no zrejme rýchlo prídu na zmysel ľudovej múdrosti, že na dobré sa zvyká rýchlo.

Visual C# Na otázku, či je programovací jazyk Visual C# úplne nový, alebo je to len modernizované objektové Céčko, (C++) môžeme povedať, že C# koncepčne vychádza z C++ a jeho tvorcovia hovoria, že všetky dobré a pokrokové črty jazyka C++ zostali zachované. Zanikli však niektoré nepopulárne črty jazyka, napríklad už spomínané pointre. Namiesto pointrov sa v C# používajú odkazy. Programátorom uľahčia život aj ďalšie črty, hlavne Garbage Collection, čo znamená, že sa už nemusíme starať o upratovanie nepotrebných objektov z pamäti. Zdalo by sa teda, že pátranie po predkovi programovacieho jazyka C# má jednoznačné riešenie, ale nie je tomu úplne tak. Je to možno paradox, ale C# má bližšie skôr k programovaciemu jazyku Java, ako k C++. Jazyk C# teda prevzal výhodné črty z viacerých „klasických“ programovacích jazykov, najviac azda z Javy, C++ a používateľské rozhranie vývojového prostredia z Visual Basicu. Presne ako vyplýva s jeho názvu (hudobníci vedia, že symbol C# znamená notu C zvýšenú o poltón), mal by tento programovací jazyk predstavovať akúsi, hudobnou terminológiou povedané „o poltón zvýšenú“ formu než spomínané programovacie jazyky.

Na obrázku vidíme príklad nasadenia aplikácie (napríklad dotazníka) na najrôznejších typoch mobilných zariadení v spolupráci s webovým a databázovým serverom.

Obrázok – Aplikačné prostredie pre mobilné zariadenia.

V ďalšej časti kapitoly ukážeme vývoj jednoduchej aplikácie v tomto vývojovom prostredí. Nezávisle na voľbe programovacieho jazyka nám sprievodca vytvorením aplikácie vytvorí prázdny aplikačný formulár, prostredníctvom ktorého môžeme navrhnúť vzhľad okna aplikácie pre komunikáciu s používateľom.



Obrázok – Pracovná plocha vývojového prostredia

Základnú konfiguráciu pracovnej obrazovky vývojového prostredia vidíme na obrázku. Pozornosť si zaslúžia hlavne panely

- Toolbox
- Solution Explorer
- Properties
- Output

Každý z týchto panelov sa obvykle skladá z niektorých záložiek, ktoré menia jeho aktuálny význam, čím je veľmi efektívne využitá zobrazovacia plocha monitora.

Toolbox

V tomto paneli na ľavej strane pracovnej ploche je dôležiá záložka **Device Controls**, ktorá obsahuje komponenty **Pointer, Label, Button, TextBox, MainMenu, CheckBox, RadioButton, PictureBox, Panel, ListBox, ComboBox, ListView, TabControl, HScrollBar, VScrollBar, Timer, DomainUpDown, NumericUpDown, TrackBar, ProgressBar, a StatusBar.**

Obrázok – Toolbox

Komponenty metódou drag and drop presúvame na plochu aplikačného formulára Niektoré z týchto komponentov sú vizuálne, to znamená, že budú v okne aplikácie viditeľné, napríklad Button, CheckBox, iné sú nevizuálne, o znamená, že ich vidíme len počas návrhu (a môžeme nastavovať ich vlastnosti). Počas behu aplikácie budú pracovať, ale na ploche aplikácie ich neuvidíme. Typickým príkladom takejto nevizuálnej komponenty je napríklad Timer. Celkovo máme v SDE k dispozícii máme tieto komponenty (v abecednom poradí).

Button
CheckBox
ComboBox
Control
ImageList
Form
Label
ListBox
ListView
MainMenu
NumericUpDown
Panel
PictureBox
ProgressBar
RadioButton
HScrollBar
StatusBar
TabControl
TextBox
Timer
TrackBar
VScrollBar

Ak tieto komponety zoradíme do skupín podľa oblasti použitia, dostaneme nasledujúci prehľad:

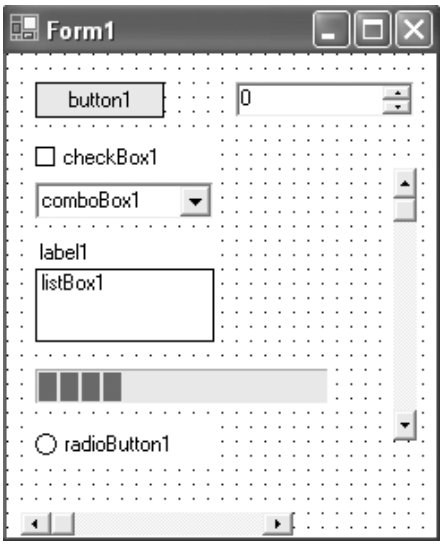
Vstup údajov
- Check Box
- Combo Box
- Command Button
- Domain Up/Down
- List Box
- Numeric Up/Down
- Radio Button
- Text Box
- Track Bar

Zobrazenie
- Label
- Picture Box
- Progress Bar
- Status Bar

Organizácia komponetov vo formulári
- List View
- Panel
- Tab Control

Pomocné komponenty
- Context Menu
- Scroll Bars
- Image List
- Main Menu
- Open & Save File Dialogs
- Timer
- Tool Bar

Viac než vysvetlenie určite napovie jednoduchý obrázok s príkladmi najčastejšie používaných vizuálnych komponentov pre vytvorenie aplikačného formulára.



Obrázok – Príklady prvkov formulára

Návod na použitie je jednoduchý. Potrebné komponenty metódou drag -and - drop premiestnime na plochu formulára a nastavíme ich vlastnosti.

Solution Explorer

Panel Solution Explorer (nazvali sme ho podľa prvej záložky) sa štandardne nachádza v pravej hornej časti okna aplikácie a pozostáva zo záložiek, ktoré vlastne prepínajú význam celého panelu

- **Solution Explorer**
- **Class View**
- **Resource View**
- **Contenst (nápovedy)**

Properties

Panel sa štandardne nachádza v pravej strednej časti okna aplikácie (pod panelom Solution Explorer) a pozostáva zo záložiek

- **Properties**
- **Dynamic Help**

V paneli Properties sa zobrazujú a samozrejme dajú editovať parametre aktívnej komponenty (t.j tej, na ktorú bolo naposledy kliknuté kurzorom myši)

Output

Panel sa štandardne nachádza v dolnej časti okna aplikácie a slúži hlavne pre zobrazenie oznamov vývojového prostredia programátorovi. Určite sa pri prvých pokusoch nevyhneme tomu, že spočiatku pôjde hlavne o chybové hlásenia :–) .

Vývoj prvej aplikácie typu Hello Word – HelloSDA

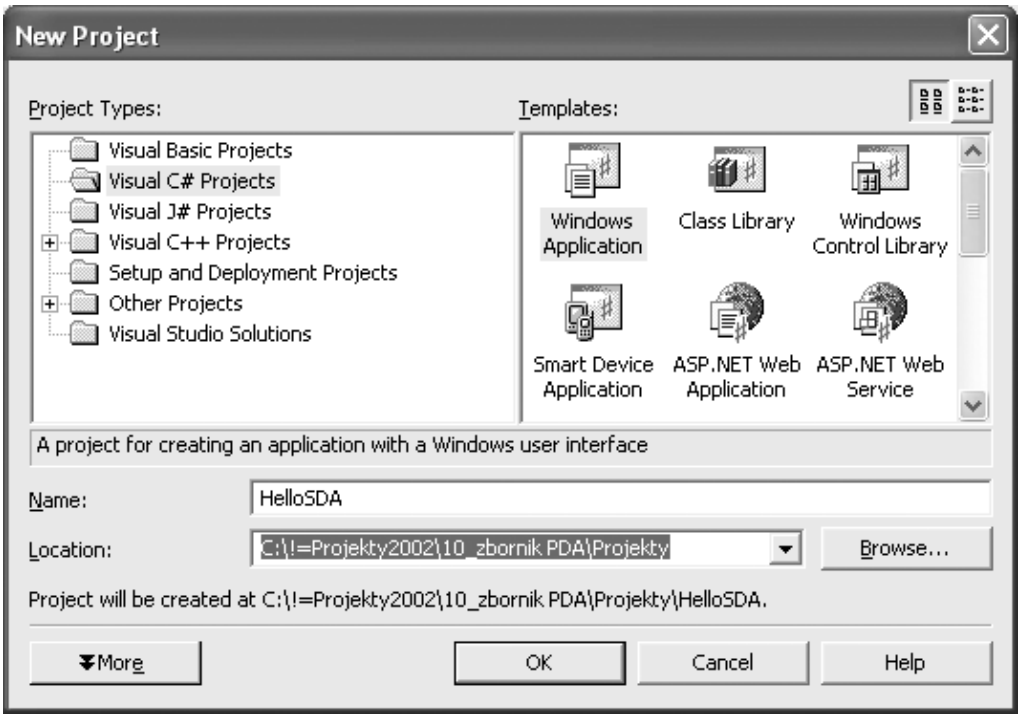
Ani v tomto prípade neporušíme tradíciu a ako prvý cvičný príklad vytvoríme aplikáciu typu Hello World. Podobne ako v predchádzajúcej kapitole okrem výpisu textu na obrazovku vytvoríme aj jednoduchý formulár, prostredníctvom ktorého bude naša aplikácia komunikovať aj s používateľom. Tento ktorý musí prostredníctvom jednoduchého dialógu zadať svoje meno (reťazec) a vek (číslo). Aj v tomto cvičnom príklade pôjde hlavne o kompletný postup napísania aplikácie a jej odladenie na softvérovom emulátore na PC a následne na jej nainštalovanie a spustenie na príslušnom mobilnom zariadení

Zadanie úlohy číslo 3.1: Vytvorte aplikáciu, ktorá umožní používateľovi zadať meno a vek a následne tieto údaje vypíše na obrazovke.

Riešenie úlohy číslo 3.1: V tejto aplikácii ukážeme riešenie zadaného problému programovacom jazyku Visual C#.

Krok 1. – Vytvorenie aplikácie typu Smart Device Application

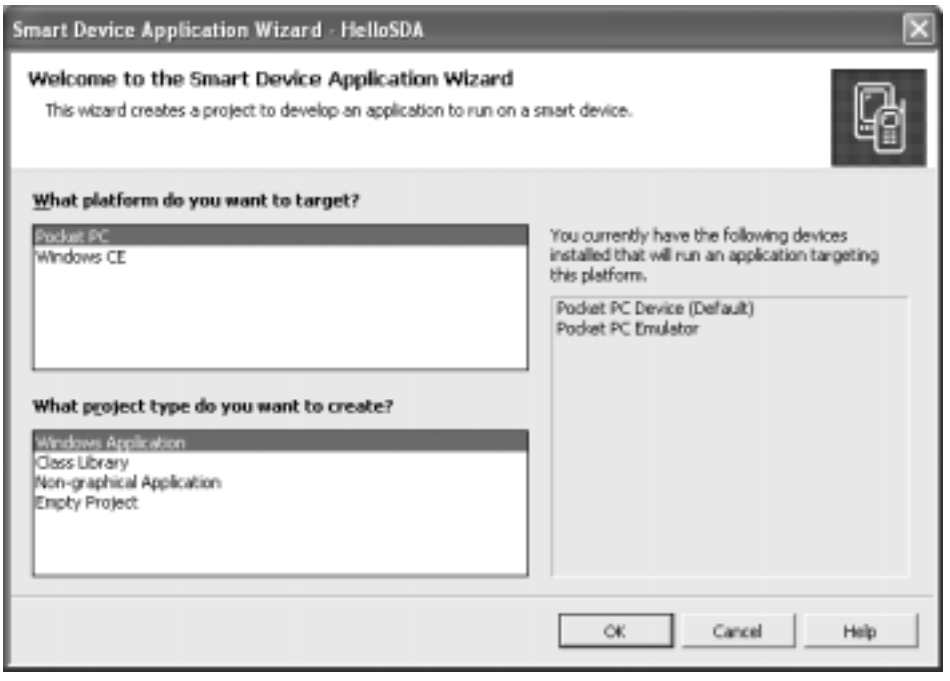
Pomocou menu **File | New | Project** vytvoríme nový projekt typu **Smart Device Application**. Podľa toho, v akom programovacom jazyku chceme náš projekt riešiť, vyberieme v okne **Templates** buď zložku **Visual Basic Projects** alebo **Visual C# Projects**. Napokon v úvodnom dialógu pomenujeme svoj projekt, (v našom prípade HelloSDA) a špecifikujeme adresár, do ktorého budú súbory tvoriace projekt uložené.



Obrázok – Nový projekt – aplikácia typu Smart Device Application

Po zatlačení tlačidla OK v ďalšom dialógu môžeme vybrať platformu **Windows CE** alebo **Pocket PC** a typ projektu. Na výber máme možnosti:

- Windows Application
- Class Library
- Non-graphical Application
- Empty Project



Obrázok – Nový projekt – Windows Application pre platformu Pocket PC

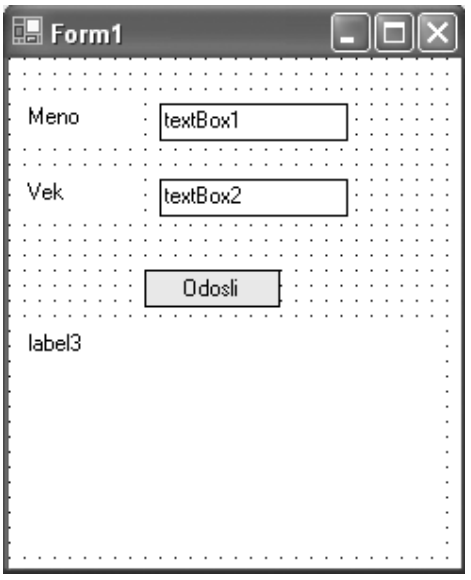
V našom prípade sme zvolili platformu Pocket PC a typ aplikácie **Windows Application**. Po potvrdení údajov zadaných v tomto dialógu nám sprievodca vytvorením aplikácie vygeneruje prázdny formulár a príslušné kódy.

Ak by našou úlohou bolo napísať len text, nemuseli by sme napísať ani jeden jediný riadok programu. Stačí len do pripraveného vizuálneho formulára vložiť komponentu Label a priradiť jej požadovaný text. Naša úloha je mierne zložitejšia a tak sa predtým, než navrhneme aplikačný formulár pozrime na kód, ktorý vygeneroval sprievodca vytvorením aplikácie. Jadro kódu je v našom prípade v súbore **Form1.cs**.

```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;

namespace HelloSDA
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
      private System.Windows.Forms.MainMenu mainMenu1;

      public Form1()
      {
        //
        // Required for Windows Form Designer support
        //
        InitializeComponent();

        //
        // TODO: Add any constructor code after InitializeComponent call
        //
      }
      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      protected override void Dispose( bool disposing )
      {
        base.Dispose( disposing );
      }
      #region Windows Form Designer generated code
      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
        this.mainMenu1 = new System.Windows.Forms.MainMenu();
        this.Menu = this.mainMenu1;
        this.Text = "Form1";

      }
      #endregion

      /// <summary>
      /// The main entry point for the application.
      /// </summary>

      static void Main()
      {
```

```
        Application.Run(new Form1());
      }
    }
}
```

Po pridaní príslušných komponentov na plochu formulára aplikácie, bude mat tento formulár (teda aj výsledná aplikácia) takýto vzhľad



Obrázok - Návrhový formulár aplikácie

Pridané komponenty boli zakomponované aj do kódu aplikácie

```
    public class Form1 : System.Windows.Forms.Form
    {
      private System.Windows.Forms.Label label1;
      private System.Windows.Forms.TextBox textBox1;
      private System.Windows.Forms.TextBox textBox2;
      private System.Windows.Forms.Label label2;
      private System.Windows.Forms.Button button1;
      private System.Windows.Forms.Label label3;
      private System.Windows.Forms.MainMenu mainMenu1;
...
```

Inicializačné parametre každej komponenty sú v procedúre **`InitializeComponent()`**

```
private void InitializeComponent()
{
        //
    // label1
    //
        this.label1.Location = new System.Drawing.Point(8, 24);
        this.label1.Size = new System.Drawing.Size(64, 20);
        this.label1.Text = "Meno";
    //
    // textBox1
    //
        this.textBox1.Location = new System.Drawing.Point(80, 24);
        this.textBox1.Text = "textBox1";
...
```

Po zostavení aplikácie v tejto etape by sa síce spustila, ale zatiaľ by nepracovala, chýba nám obsluha tlačidla. Jeho funkcia bude jednoduchá. Po zatlačení sa v komponente **label3** zobrazí text zadaný v komponentoch **textBox1** (meno) a **textBox2** (vek). U zložitejšej aplikácie by sa určite oplatilo premenovať implicitné názvy komponentov napríklad na tbMeno a tbText. Ak sa pozrieme na návrhový formulár, určite by sa nám vo výslednej aplikácii nepáčilo, že v editačných oknách by sa implicitne zobrazil text **textBox1**, **textBox2** a **label3**. Preto v paneli Properties vymažeme príslušné texty z položky Text (Pozor, nie z položky Name)

Procedúra pre obsluhu kliknutia na tlačidlo Odošli je zatiaľ prázdna.

```
private void button1_Click(object sender, System.EventArgs e)
{

}
```

Pridáme do nej jednoduchý kód, ktorý zostaví výsledný textový reťazec a zobrazí ho pomocou komponenty **label3**:

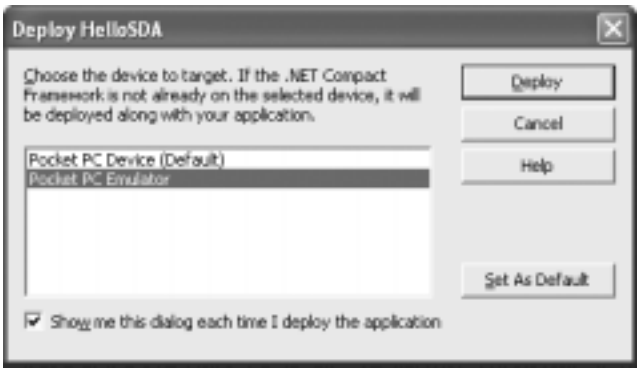
```
private void button1_Click(object sender, System.EventArgs e)
{
    string sOznam;
    sOznam = "Meno: "+ textBox1.Text + " \r\nVek: " + textBox2.Text;
    label3.Text = sOznam;
}
```

Našim ďalším cieľom je preložiť aplikáciu a spustiť ju na emulátore. V hlavnom menu aktivujeme voľby **Build | Build HelloSDA** . V spodnom okne vývojového prostredia potom môžeme sledovať priebeh prekladu našej aplikácie.

```
------ Build started: Project: HelloSDA, Configuration: Debug Pocket PC ------

Preparing resources...
Updating references...
Performing main compilation...

Build complete -- 0 errors, 0 warnings
Building satellite assemblies...
Visual Studio is ready to deploy HelloSDA

---------------------- Done ----------------------
    Build: 1 succeeded, 0 failed, 0 skipped
```

Pomocou ovládacieho prvku vývojového prostredia (umiestneného nad toolboxom) zvolíme, či chceme spustiť našu aplikáciu pomocou emulátora, alebo ju nainštalovať do reálneho mobilného zariadenia. Prvé ladiace úkony urobíme spravidla za pomoci emulátora, preto nastavíme voľbu **Pocket PC Emulátor**.

V hlavnom menu aktivujeme voľby **Build | Deploy HelloSDA**. Pomocou dialógového okana Deploy.. sa ešte stále môžeme pomocou akých prostriedkov budeme ladiť našu aplikáciu.



Obrázok – Dialóg Deploy

V spodnom okne vývojového prostredia potom môžeme sledovať priebeh zostavovania a prenosu našej aplikácie do emulátora

```
------ Deploy started: Project: HelloSDA, Configuration: Debug Pocket PC ------
Deploying to Pocket PC Emulator using Emulation Transport
Connected to Pocket PC Emulator (Pocket PC) running on X86.
Copying files from 'C:\!=Projekty2002\10_zbornik PDA\Projekty\HelloSDA\bin\Debug' to '\Program
Files\HelloSDA'
Copying HelloSDA.exe
Copying netcf.core.ppc3.x86.cab
Launching device installation of 'netcf.core.ppc3.x86.cab'. Please check device screen for
further instructions......
Copying System_SR_enu.cab
Launching device installation of 'System_SR_enu.cab'. Please check device screen for further
instructions..

---------------------- Done ----------------------
    Build: 1 succeeded, 0 failed, 0 skipped
    Deploy: 1 succeeded, 0 failed, 0 skipped
```

Po spustení môžeme pomocou emulátora vyskúšať funkčnosť našej aplikácie.

Obrázok – Spustenie aplikácie pomocou emulátora

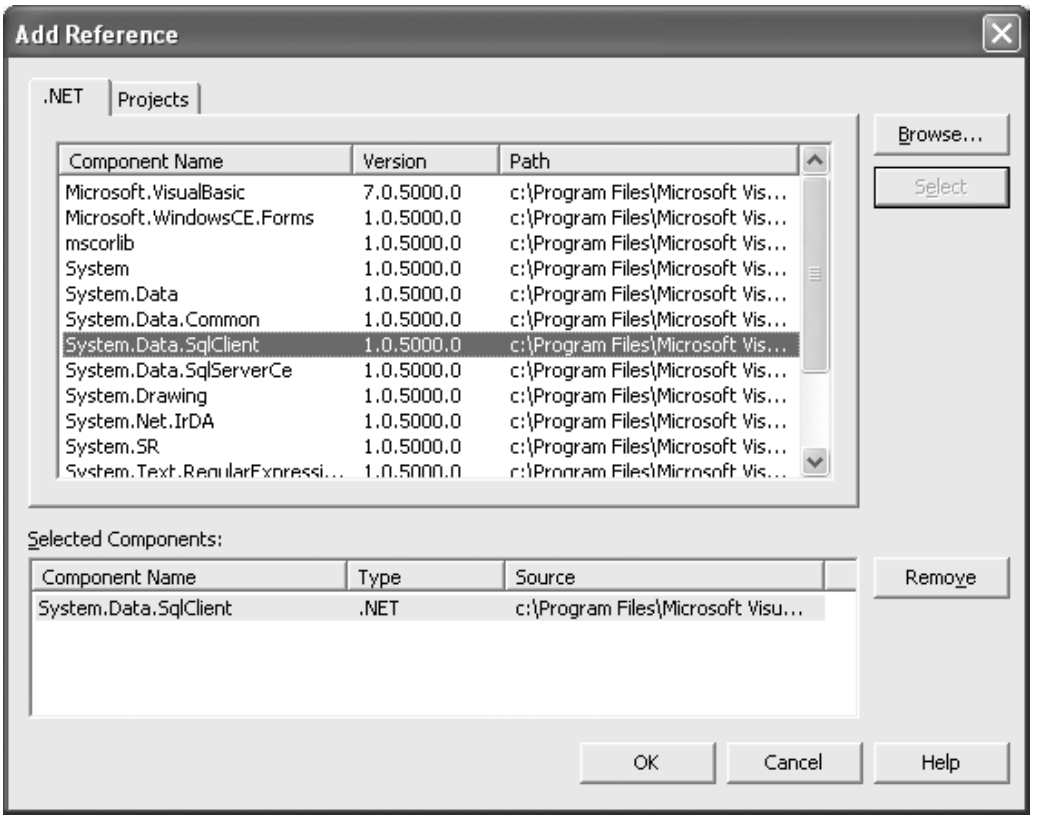
Týmto sme vlastne do bodky splnili literu zadania cvičnej aplikácie.

Databázová aplikácia využívajúca SqlClient managed provider

Aplikácia tohoto typu využíva skutočnosti, že vrstva Compact Framevork má integrovanú komponentu SqlClient managed provider. Pomocou tejto komponenty je možné pristupovať k údajom pod správou databázových serverov Microsoft SQL Server 2000, prípadne staršej verzie MS SQL Server 7.0.

Zadanie úlohy číslo 3.2: Vytvorme jednoduchú klientskú databázovú aplikáciu, ktorá bude využívať SqlClient managed provider. Podmienkou pre činnosť takejto aplikácie je pochopiteľne trvalé spojenie na SQL Server 2000
.
Riešenie úlohy číslo 3.2: Vo Visual Studiu .NET vytvoríme nový projekt typu **Smart Device Application**. v jazyku **Visual C#**. Projekt nazveme napríklad KlientSQL.

V okne Solution explorer pomocou volby Add Reference pridáme referencie na komponenty **System.Data.SqlClient a System.Data.Common**.

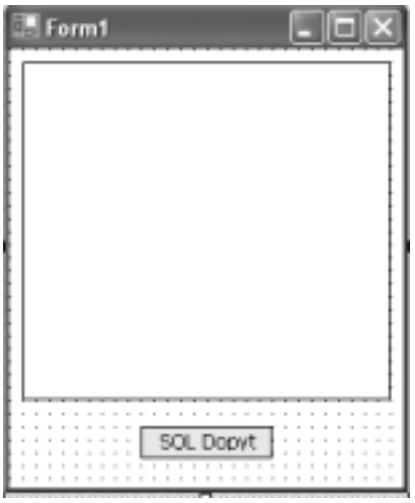


Obrázok – vytvorenie referencie na System.Data.SQLClient

Ak nemáme vytvorené prepojenie na SQL Server, musíme si ho vytvoriť. Podrobnejší postup aj s obrázkom je v prílohe venovanej vytvoreniu webovej služby na konci prvej kapitoly. Takže len heslovito. Ľavé okno prepneme do režimu **Server Explorer** a vytvoríme nové pripojenie v zložke **Data Connections**. (pravé tlačidlo, voľba **Add Connection**). Po vytvorení prepojenia presunieme jeho ikonu na pracovnú plochu aplikácie pod aplikačný formulár. Na začiatok zdrojového súboru Form1.cs pridáme kód

```
using System.Data.SqlClient;
using System.Data.Common;
```

Návrh aplikačného formulára je v tomto prípade veľmi jednoduchú a pozostáva len z dvoch komponentov. Komponenty List View s názvom lvVysledok a tlačidla s názvom btSQL.



Obrázok – Návrh aplikačného formulára

Celé dejstvo sa potom bude odohrávať v obslužnej procedúra tlačidla `btSQL_Click`.

```
private DataTable dtEmp;

private void btSQL_Click(object sender, System.EventArgs e)
{

    SqlCommand cmd = new SqlCommand("SELECT LastName, FirstName, Title, Address, City, Country,
Notes FROM Employees where EmployeeID = @EMPID", sqlConnection1);
    cmd.Parameters.Add("@EMPID", 2);
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    DataSet ds = new DataSet();
    da.Fill(ds, "Employees");
...
//kod pre naplnenie komponenty ListView
...
  foreach(DataRow row in dtEmp.Rows)
  {
...
  }

}
```

Všimnime si pri prvom zavádzaní tohoto typu aplikácie sa do mobilného zariadenia inštaluje Microsoft SQL Klient

Obrázok – Návrh aplikačného formulára

Podrobný kód pre naplnenie údajov do komponenty List View je v nasledujúcom, z hľadiska použitej architektúry oveľa perspektívnejšom príklade.

Komunikácia s databázou pomocou webových služieb

Vzhľadom na určité špecifiká platformy Pocket PC 2002, ktorá určite nie je určená ani na uschovávanie údajov v databázach rádu stoviek megabajtov, nehovoriac už o gigabajtových, či terabajtových databázach, pracujeme s takýmito databázami pomocou aplikácií, ktoré vystupujú ako klient voči databázovému serveru. V tomto prípade využijeme údaje poskytované webovou službou vo formáte XML.

Zadanie úlohy číslo 3.3: V ďalšej cvičnej aplikácii vytvoríme jednoduchý prehliadač údajov uložených v databáze. K údajom sa napojíme prostredníctvom XML webovej služby. Úlohou aplikácie bude na základe zadaného názvu mesta vypísať z neho všetkých zákazníkov.

Riešenie úlohy číslo 3.3: Využijeme webovú službu **wsZamestnanci**, ktorá je popísaná v prílohe k prvej kapitole. Pripomeňme si jej metódu **`DajZakaznikovZmesta`**, ktorá vyžaduje ako parameter reťazec s požadovaným mestom. Metóda vráti objekt typu Dataset.

```
[WebMethod]
public DataSet DajZakaznikovZmesta(string sCity)
{
   SqlCommand cmd = new SqlCommand("SELECT ContactName, Address FROM Customers WHERE City=
@PCITY", sqlConnection1);
   cmd.Parameters.Add("@PCITY", sCity);
   SqlDataAdapter da = new SqlDataAdapter(cmd);
   DataSet ds = new DataSet();
   da.Fill(ds, "Employees");
   sqlConnection1.Close();
   return ds;
}
```

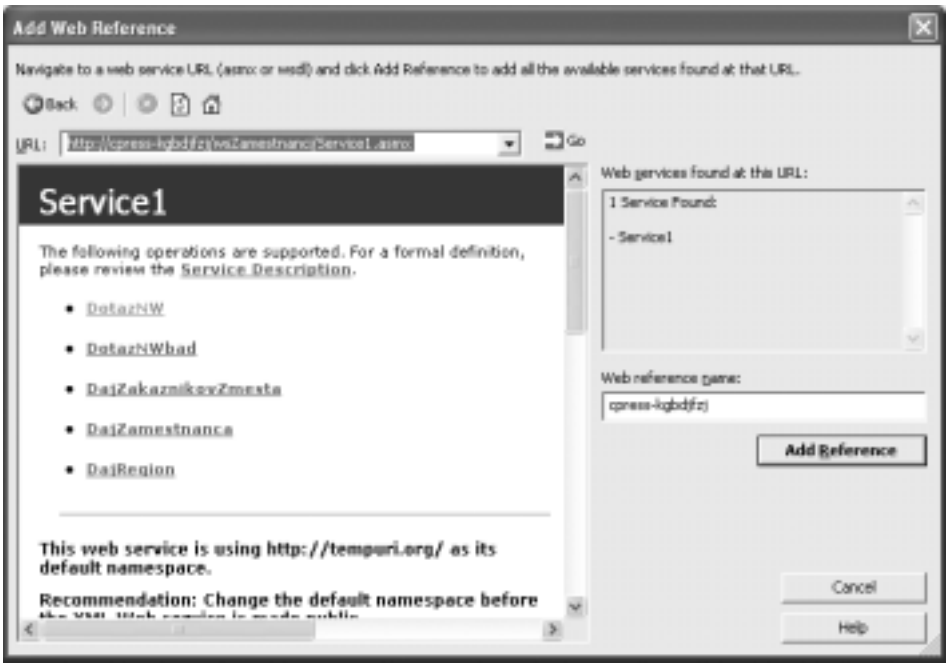
Pre riešenie využijeme Visual Studio.NET 2003 (Everett), prípadne Visual Studio .NET s nainštalovaným rozšírením **SDE**.

Vytvorenie aplikácie typu Smart Device Application

Vráťme sa k cvičeniu. Vytvoríme nový projekt z názvom napríklad **ZakazniciSDA** typu **Smart Device Application** v zložke **Visual Basic Visual C# Projects**.
Navrhneme dialóg aplikácie. Bude obsahovať komponentu Label s textom „Mesto“, komponentu TextBox s názvom tbMesto a tlačidlo btDaj s textom „Daj“. Dominantnou komponentou aplikačného formulára bude komponenta ListView s názvom lvZakaznici.

Obrázok – Návrh formulára aplikácie

V nasledujúcom kroku zaregistrujeme XML webovú službu. Klikneme pravým tlačidlom myši na položku **References** v okne **Solution Explorer** a v zobrazenom menu klikneme na položku **Add Web Reference**. V dialógu, ktorý sa následne otvorí zadáme referenciu na webovú službu, v našom prípade adresu *http://CPRESS-KGBDJFZJ/wsZamestnanci/Service1.asmx* Tlačidlom Add Reference pridáme do projektu referenciu na webovú službu



Obrázok – Pridanie referencie na webovú službu

Referenciu v pravom okne vývojového prostredia premenujeme na wsZamestnanci. Ak sa pozrieme do adresára nášho projektu, nájdeme tam podadresár Web References | wsZamestnanci, ktorý obsahuje niekoľko súborov obsahujúce proxy triedy a interakcie SOAP.

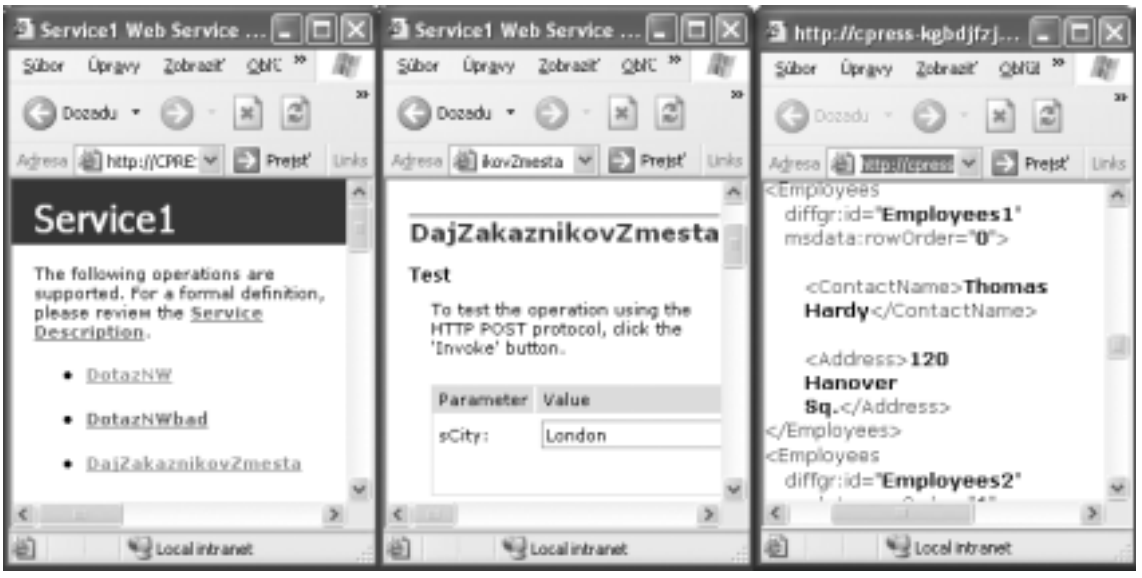
Pri vývoji databázovej aplikácie býva dobrým zvykom odladiť si SQL príkaz pomocou konzolovej aplikácie databázového servera, napríklad pomocou SQL Query analyzera. V tomto prípade, keďže využívame webovú službu, už značná časť databázovej logiky bola odladená v procese jej vývoja , no nezaškodí pripomenúť si staré dobré zásady. V prvom priblížení zostavíme SQL dotaz „natvrdo“ pre mesto London.

```
SELECT ContactName, Address FROM Customers WHERE City='London'
```

Vidíme, že tento SQL dotaz je nielen syntakticky správny, ale aj vracia niekoľko záznamov

```
ContactName                    Address
------------------------------ ------------------------------------
Thomas Hardy                    120 Hanover Sq.
Victoria Ashworth              Fauntleroy Circus
Elizabeth Brown                Berkeley Gardens 12  Brewery
Ann Devon                       35 King George
Simon Crowther                 South House 300 Queensbridge
Hari Kumar                      90 Wadhurst Rd.

(6 row(s) affected)
```

Rovnako otestujeme funkčnosť webovej služby



Obrázok – Test webovej služby

Na začiatok zdrojového súboru **Form1.cs** pridáme referenciu

```
using System.Net;
```

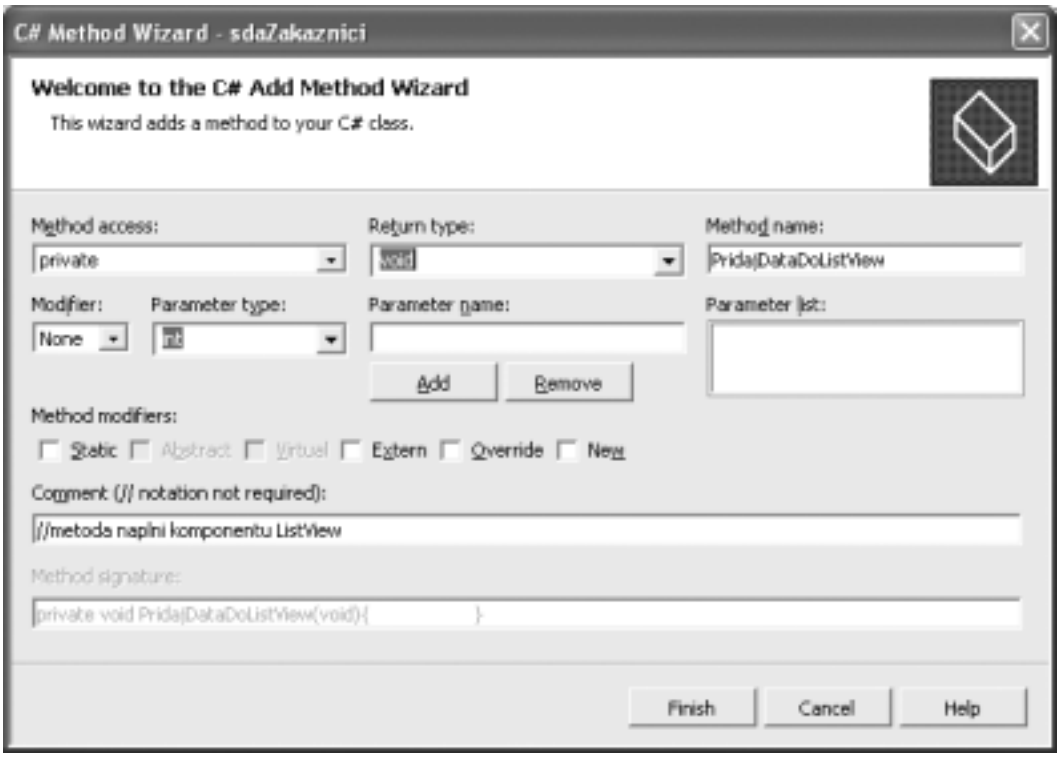
a inštancie proxy tried webovej služby

```
private DataSet dsZakaznici;
private DataTable dtCustomers;
private wsZamestnanci.Service1 ws = new wsZamestnanci.Service1();
```

Jadro kódu bude v procedúre **`btDaj_Click`**. Všimnime si, že v praktickej aplikácii je dobrým zvykom poistiť sa proti vzniku chýb a výnimiek (programová konštrukcie try – catch).

```
private void btDaj_Click(object sender, System.EventArgs e)
{
    try
    {
      dsZakaznici = ws.DajZakaznikovZmesta (tbMesto.Text);
      dtCustomers = dsZakaznici.Tables["Northwind"];
      PridajDataDoListView();
    }
    catch (WebException we)
    {
        MessageBox.Show("Nemozem sa pripojit. Chyba: " + we.Message,
                    "Connection Failed");
    }

}
```

Jednotlivé záznamy zapúzdrené v datasete pridáme do komponenty ListView v pomocou novovytvorenej metódy **`PridajDataDoListView()`**
Novú metódu pridáme napríklad pomocou dialógu **C# Method Wizard**, ktorý sa aktivuje v kontextovom menu po kliknutí ľavým tlačidlom myši na názov triedy Form1.



Obrázok – C# Method Wizard

Kód v tele procedúry je pomerne jednoduchý, jedná sa o pridanie obsahu dvoch stĺpcov datasetu do komponenty ListView.

```
private void PridajDataDoListView()
{
    ListViewItem polozka;
    lvZakaznici.Clear();
    lvZakaznici.Columns.Add("ContactName",(lvZakaznici.Width/2), HorizontalAlignment.Left);
    lvZakaznici.Columns.Add("Address", (lvZakaznici.Width/2),HorizontalAlignment.Right);
    lvZakaznici.View = View.Details;

    foreach(DataRow row in dtCustomers.Rows)
    {
       polozka = new ListViewItem(row["ContactName"].ToString());
       polozka.SubItems.Add(row["Address"].ToString());
       lvZakaznici.Items.Add(polozka);
    }

}
```

Výsledok v okne emulátora na náš testovací dotaz svedčí o jeho správnosti.

Obrázok – Test aplikácie

Už v závere predchádzajúceho príkladu sme naznačili, že riešenie s využitím webovej služby je oveľa čistejšie z hľadiska použitej architektúry. Formátové konverzie údajov môžu nepatrne znížiť výkon, ale táto drobná nevýhoda je vyvážená jednoduchosťou klientskej aplikácie, ale hlavne menšou frekvenciou nutnosti upgrade u klienta. Vo väčšine príkladov postačí úprava webovej služby a takto inovovaná aplikačná logika je bez nutnosti akýchkoľvek zásahov u klienta ihneď k dispozícii. Výhody z hľadiska jednoduchosti a čistoty oddelenia jednotlivých architektonických vrstiev aplikácie sú taktiež neoddiskutovateľné. K webovej službe môžeme cez HTTP protokol pristupovať prakticky odkiaľkoľvek a môžeme je využívať aj z iných typov aplikácií (podrobnejšie popísané v kapitole 5 venovanej MMIT).

Kapitola 4

Replikácia údajov v databázach, SQL Server CE 2.0

Pri implementovaní databázového servera na určitú hardvérovú platformu nás zaujímajú možnosti tejto platformy, hlavne úložná a pamäťová kapacita a výpočtový výkon. U platformy PocketPC to s výkonom procesora vyzerá celkom slušne, od typických 200MHz sa takt procesora pohybuje smerom hore (napríklad Toshiba e740 má procesor taktovaný na 400 MHz). Horšie je to s kapacitou úložiska pre údaje. Mobilné zariadenia spravidla nedisponujú pevným diskom a pamäťová kapacita sa pohybuje v rozsahu 32 – 64 MB. V porovnaní s terabajtovými klastrovými servermi žalostne málo, ale 32 megabajtov predstavuje 32 miliónov znakov, a kým toľkoto údajov získame a ručne uložíme, alebo naopak vo forme hypertextových výpisov odprezentujeme zákazníkovi, nejaké mesiace nám aj takáto pamäť postačí. Samozrejme záleží od formy údajov. V prípade multimediálnych údajov sa mesiace môžu zmeniť na minúty. Mobilné zariadenia disponujú však aj pamäťovými kartami a do slotu Compact Flash II môžeme zasunúť napríklad pamäťové médium IBM Microdrive s kapacitou 1GB

Konfigurácia, kedy je databáza aj jadro databázového servera umiestnená priamo v mobilnom zariadení je však len jednou s možností. Komunikačné možnosti týchto zariadení priam ponúkajú alternatívu, kedy sú údaje uložené v databázovom serveri a z mobilného zariadenia sa k nim len pristupuje ako z klientskej aplikácie.

Typické oblasti nasadenia a použitia mobilných databáz:
– sklad vybavený sieťou wireless LAN
– zber údajov v teréne
– pracovník zásielkovej služby
– pracovník marketingu na pracovnej ceste

Mobilná databázová aplikácia môže pracovať v dvoch režimoch
– on -line
– off -line

On - line

Pri tomto type aplikácie sa predpokladá trvalé pripojenie PDA do siete. Typický príklad on-line databázovej aplikácie je aplikácia typu „elektronický obchod“. Predstavme si napríklad zásielkový predaj počítačov a ich komponentov. Zákazník si prostredníctvom Internetu objedná tovar a zvyšok už musí zariadiť zásielková firma. Okrem dátových transakcií musíme tovar v sklade nájsť a vyskladniť, prípadne zákazníkovi operatívne poskytnúť informáciu, že požadovaný tovar momentálne na sklade nemáme. Je to pomerne náročná úloha. Ak však v budovách firmy a skladu zriadime rádiovú sieť, môžeme priamo pomocou PDA zadávať úlohy jednotlivým skladníkom a kontrolovať potvrdenie vyskladnenia tovaru.

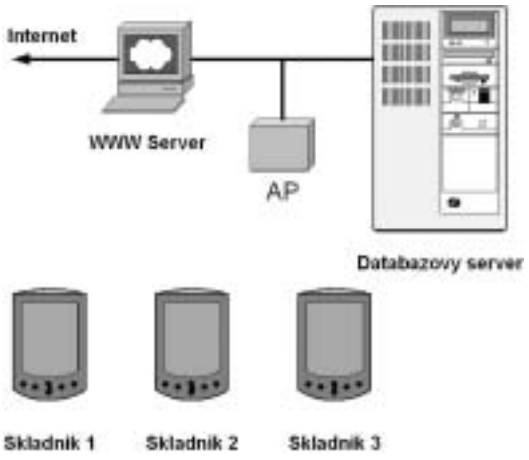
Typická architektúra internetového obchodu pozostáva dvoch vrstiev:
- vrstva **Front - end** rieši interakciu so zákazníkov
- vrstva **Back - end** rieši komunikáciu a dátové toky vo vnútri firmy tak, aby bola objednávka zákazníka promptne a spoľahlivo vybavená.

Rádiová sieť podstatne zvýši efektivitu vrstvy Back -end. Táto vrstva sa skladá zo serverových, transakčných a databázových komponentov.
Pre názornosť uvedieme chronologický popis takejto transakcie.
- Zákazník si na základe ponuky vyberie a objedná tovar.
- Požiadavku zákazníka spracuje webový server v spolupráci s databázovým serverom. Overí sa identita zákazníka a spôsob úhrady za tovar.
- Na základe údajov z databázy o stave zásob tovaru na sklade sa objednávka bud' príjme, alebo zamietne.
- Potom dostane skladník pokyn k vyskladneniu požadovaného tovaru. Úspešnosť tejto operácie skladník potvrdí do centra transakčnému serveru. Nakoniec dostane zákazník vyrozumenie o spôsobe doručenia tovaru

Použitá technológia

Údaje o tovare v sklade môžu byť snímané čítačkou čiarových kódov pripojenou alebo dokonca integrovanou k PDA, spracovávané na serveri a ukladané napríklad v databáze spravovanej SQL Serverom 2000.
V priestoroch skladu sú inštalované prístupové body **Access Pointy**, ktoré pokrývajú sklad rádiovým signálom a zabezpečujú bezdrôtovú konektivitu mobilných terminálov s pevnou sieťou. Počet prístupových bodov je daný miestnymi podmienkami – počet uličiek, výška regálov atď. V priestore pokrytom rádiovým signálom sa môžu voľne pohybovať pracovníci alebo dopravné a manipulačné prostriedky vybavené terminálmi s integrovaným snímačom čiarového kódu. Mobilné terminály sa automaticky prihlasujú k prístupovému bodu s najvhodnejšími prijímacími

podmienkami na báze **roamingu**, podobne ako pri mobilných telefónoch. Zosnímané dáta sú prenášané on-line do počítačovej siete LAN. Pomocou aplikačného softvéru sú tieto informácie spracované na a uložené v databáze na databázovom serveri. Tento systém môže samozrejme komunikovať aj s firemnými informačnými systémami veľkoodberateľov formou výmeny potrebných údajov.



Obrázok 4.1 – On line databázová aplikácia

Off - line

Pri tomto type aplikácie sa ako typický režim predpokladá autonómna práca, s občasným pripojením do siete, buď pomocou dátového prenosu GSM, alebo mobilné zariadenie sa občas (napríklad po návrate pracovníka z pracovnej cesty) pripojí do siete. Tento režim je využiteľný pre obchodných cestujúcich, pracovníkov zásielkových služieb a podobne. Off -line aplikačný režim sa hodí aj pre zber údajov v teréne, rôzne ankety, prieskumy a podobne.

Databázová aplikácia na mobilnom zariadení

Pri návrhu koncepcie databázovej aplikácie pre mobilné zariadenia môžeme použiť niekoľko aplikačných scenárov.

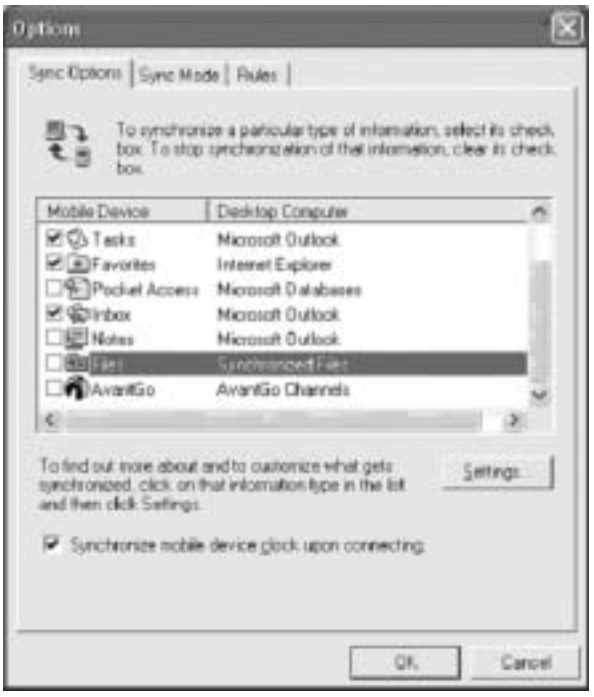
Replikácia XML súborov pomocou ActiveSync

Obrázok 4.2 – Replikácia pomocou ActiveSync

S replikáciou tohoto typu sa stretávajú používatelia mobilných zriadení pomerne často. Jednak synchronizujú súbory na ktorých pracujú, napríklad prenášajú elektronické knihy pre MSReader, dokumenty aplikácií Word – Pocket Word, prípadne synchronizujú údaje medzi aplikáciami Outlook a Pocket Outlook a podobne. Výhodou je možnosť práce off-line, len s občasným pripojením sa k desktopu, alebo serveru. Určitou nevýhodou je možnosť súčastného pripojenia len jedného mobilného zariadenia k jednému PC.

Obrázok 4.3 – Replikácia súborov pomocou ActiveSync

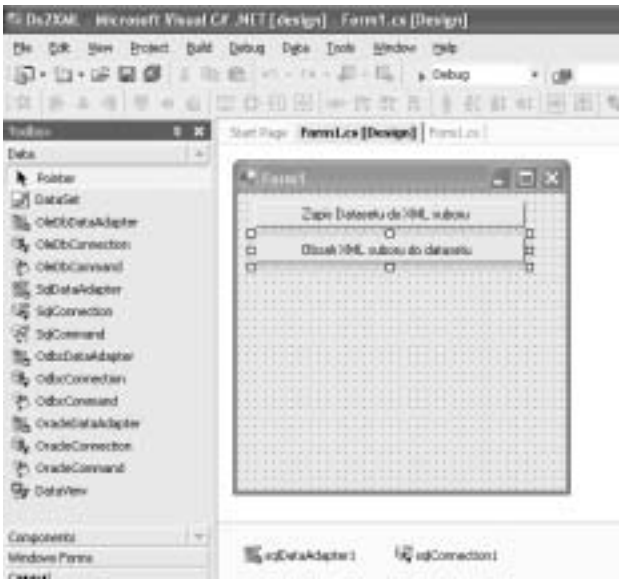
Ako vyplýva s našej schémy, postačí v manuálnom alebo poloautomatickom režime zosynchronizovať XML súbory, ktoré obsahujú príslušné údaje. V menu **Tools | Options**, záložke **Sync Options** nastavíme zložky a XML súbory, obsah ktorých sa bude synchronizovať. Môžeme nastaviť aj vlastnosti synchronizácie, teda ktoré zariadenie je voči tomu druhému, čo sa týka údajov dominantné.



Obrázok 4.4 – Nastavenie synchronizácie súborov.

Zápis datasetu do XML.

Téma „Dataset a XML“ je vyčerpávajúco popísaná v iných publikáciách aj v dokumentácii no nezaškodí si však pripomenúť aspoň základy a ukázať na jednoduchom príklade uloženie obsahu datasetu do XML súboru a jeho opätovné načítanie z XML súboru do datasetu. Pre jednoduchosť vyrobíme vo Visual Studiu .NET klasickú desktopovú Windows aplikáciu v programovacom jazyku C#. Pomocou menu **File | New | Project** vytvoríme nový projekt typu **Windows Application**. v zložke **Visual C# Projects**. Projekt pomenujeme DS2XML. Návrh aplikačného formulára tejto cvičnej miniaplikácie je veľmi jednoduchý. Postačia dva tlačidlá **Zápis Datasetu do XML súboru** a **Obsah XML súboru do datasetu**. Zo zložky Data toolboxu presunieme na plochu aplikácie komponentu sqlDataAdapter a nastavíme jeho parametre na databázu Northwind.



Obrázok 4.5 – Cvičná aplikácia pre zápis datasetu do XML súboru

Procedúra pre obsluhu tlačidla **Zápis Datasetu do XML súboru** bude využívať členskú metódu datasetu WriteXml pre zápis jeho obsahu do XML súboru.

```
private void ds_to_xml_Click(object sender, System.EventArgs e)
{
    DataSet dsRegion = new DataSet();
    sqlDataAdapter1.Fill(dsRegion, "Region");
    dsRegion.WriteXml("c:\\Region.xml", XmlWriteMode.IgnoreSchema);
}
```

O fungovaní našej procedúry sa presvedčíme veľmi jednoducho. Obsah datasetu, v našom prípade štyri záznamy sa zapíšu do XML súboru `Region.xml`

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Region>
    <RegionID>1</RegionID>
    <RegionDescription>Eastern                </RegionDescription>
  </Region>
  <Region>
    <RegionID>2</RegionID>
    <RegionDescription>Western                </RegionDescription>
  </Region>
  <Region>
    <RegionID>3</RegionID>
    <RegionDescription>Northern               </RegionDescription>
  </Region>
  <Region>
    <RegionID>4</RegionID>
    <RegionDescription>Southern               </RegionDescription>
  </Region>
</NewDataSet>
```

Procedúra pre obsluhu tlačidla **Obsah XML súboru do datasetu** bude využívať členskú metódu datasetu ReadXml pre zápis jeho obsahu do XML súboru.

```
private void xml_to_ds_Click(object sender, System.EventArgs e)
{
    DataSet mojDS = new DataSet();
    mojDS.ReadXml("c:\\Region.xml", XmlReadMode.Auto);
}
```

ADO.NET Data provider

Pri tomto spôsobe komunikácie musí byť zabezpečené trvalé spojenie medzi serverom a mobilným klientom. Výhodou je rýchly prístup prakticky k neobmedzenému množstvu údajov uložených v databáze SQL Servera. Pokiaľ využijeme tento spôsob prístupu, máme k dispozícii všetky vymoženosti poskytované „veľkým“ SQL Serverom, ako sú napríklad procedurálny jazyk T-SQL (Transact SQL), uložené procedúry a podobne. Tento spôsob je ideálny pre komunikáciu v dosahu rádiovej siete, napríklad v skladoch a podobne. Nevýhodou je dvojvrstvová architektúra na serveri a tým pádom obmedzená škálovateľnosť.

Obrázok 4.6 – ADO.NET Data Provider

V aplikáciách môžeme využiť komponentu **System.Data.SqlClient** pre priamy prístup k údajom pod správou databázového servera SQL Server 2000 (alebo verzie 7.0).

Využitie XML webovej služby

Tento scenár je popísaný v tretej kapitole. Aplikácia využíva údaje poskytované webovou službou vo formáte XML.

Obrázok 4.7 – Využitie XML webovej služby

SQL CE replikácia

Výhodou tejto konfigurácie je použitie lokálnej databázy, ktorá je vždy prístupná a možnosť komplexného dopytovania prostredníctvom jazyka SQL. Synchronizácia je realizovaná prostredníctvom HTTP protokolu. Nevýhodou je nutnosť konfigurácie servera a taktiež databázový engine zaberá v pamäti mobilného zariadenia nejaké miesto.

Obrázok 4.8 – SQL CE replikácia

Prístup k SQL Serveru cez HTTP

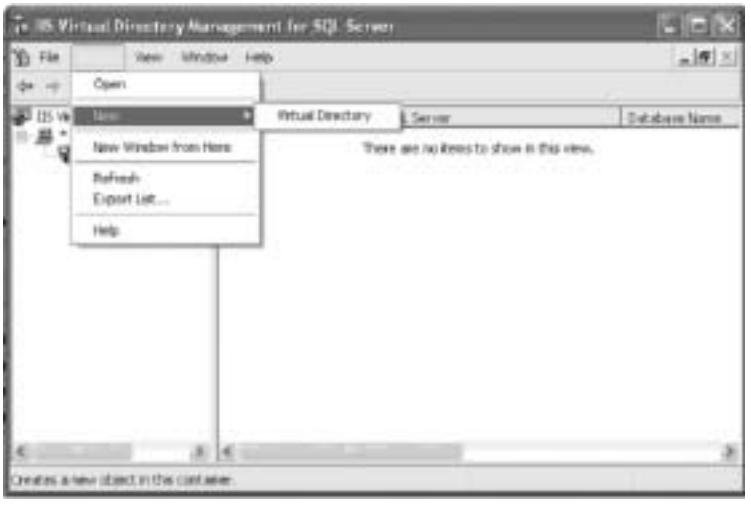
Príkaz v jazyku SQL môže byť aj súčasťou URL adresy zadanej z PC alebo mobilného zariadenia prostredníctvom prehliadača HTML stránok. SQL Server potom vygeneruje údaje vo forme XML súboru.

Aby SQL Server 2000 vygeneroval údaje vo formáte XML použijeme SQL príkaz doplnený o klauzulu **`FOR XML`**. napríklad

```
SELECT * FROM Employees FOR XML AUTO
```

Predtým než budeme môcť pristúpiť k databáze prostredníctvom SQL príkazu v URL adrese musíme nakonfigurovať Internet Information Server (IIS) pre podporu virtuálneho adresára.

Najskôr si vytvoríme adresár, napríklad **`C:\inetpub\wwwroot\nwind\Virt_adresar`**. Bude to fyzický adresár na disku ku ktorému priradíme virtuálny adresár pomocou nástroja **IIS Virtual Directory Management for SQL server**. Tento nástroj je prístupný cez položku menu SQL Servera s názvom **Configure SQL XML Support in IIS**.



Obrázok 4.9 – Vytvorenie virtuálneho adresára

Pomocou tlačidla **Action** na tlačidlovej lište programu si vytvoríme nový virtualny adresár s názvom **vd_nwind**. Tlačidlom Browse určíme cestu k fyzickému adresáru, v našom prípade k adresáru **C:\Inetpub\Wwwroot\nwind\Virt_adresár**. Povolíme služby, ktoré budeme na virtuálny adresár smerovať.
– Allow URL queries,
– Allow template queries,
– Allow XPath,
– Allow POST

Vytvoríme adresáre pre šablóny a schémy, napríklad
C:\Inetpub\Wwwroot\nwind\Virt_adresár\template
C:\Inetpub\Wwwroot\nwind\Virt_adresár\schema

Samozrejme, že musíme zaistiť prístupové privilégia (pomocou nástroja **SQL Server Enterprise Manager**) Teraz už môžeme prostredníctvom internetového prehliadača zadaním URL adresy položiť dotaz priamo SQL Serveru.

http://llhome/vd_nwind?sql=SELECT+*+FROM+Employees+FOR+XML+AUTO&root=ROOT

V okne prehliadača sa objaví výpis údajov vo formáte XML

<?xml version="1.0" encoding="utf-8" ?>
- <ROOT>
<Employees EmployeeID="**1**" LastName="**Davolio**" FirstName="**Nancy**" Title="**Sales Representative**" TitleOfCourtesy="**Ms.**" BirthDate="**1948-12-08T00:00:00**" HireDate="**1992-05-01T00:00:00**" Address="**507 - 20th Ave. E. Apt. 2A**" City="**Seattle**" Region="**WA**" PostalCode="**98122**" Country="**USA**" HomePhone="**(206) 555-9857**" Extension="**5467**" Photo="**dbobject/Employees[@EmployeeID='1']/@Photo**" Notes="**Education includes a BA in psychology from Colorado State University in 1970. She also completed "The Art of the Cold Call." Nancy is a member of Toastmasters International.**" ReportsTo="2" PhotoPath="**http://accweb/emmployees/davolio.bmp**" />
<Employees EmployeeID="**2**" LastName="**Fuller**" FirstName="**Andrew**" Title="**Vice President, Sales**" TitleOfCourtesy="**Dr.**" BirthDate="**1952-02-19T00:00:00**" HireDate="**1992-08-14T00:00:00**" Address="**908 W. Capital Way**" City="**Tacoma**" Region="**WA**" PostalCode="**98401**" Country="**USA**" HomePhone="**(206) 555-9482**" Extension="**3457**" Photo="**dbobject/Employees[@EmployeeID='2']/@Photo**" Notes="**Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager in January 1992 and to vice president of sales in March 1993. Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.**" PhotoPath="**http://accweb/emmployees/fuller.bmp**" />
...

Na tomto mieste je potrebné vysvetliť, prečo sme v URL adrese použilo namiesto medzery znak +. V tomto prípade to má len estetický význam, pretože ak zadáme URL adresu s medzerami

http://llhome/vd_zak?sql=SELECT * FROM Employees FOR XML AUTO&root=ROOT
vidíme, že po odolaní URL adresy sa reťazec v okne adresy zmenil na tvar:

http://llhome/vd_zak?sql=SELECT%20*%20FROM%20Employees%20FOR%20XML%20AUTO&root=ROOT

pretože URL adresa nesmie obsahovať medzery. Prehliadač vypísal výsledok, ale nahradil všetky medzery v adrese ich hexadecimálnym kódom %20. Niektoré znaky v URL adrese majú špeciálny význam. Napríklad znak ? sa používa v URL adrese ako oddeľovač parametrov, v SQL príkaze SELECT, konkrétne v klauzuli WHERE sa používa ako zástupný znak, alebo, podobne ako v našom prípade niektorý stĺpec môže priamo obsahovať hodnotu ?.Preto namiesto znakov, ktoré majú v URL adrese špeciálny význam, používame zástupné znaky. Namiesto medzery (hexadecimálne %20) používame znak +, namiesto otáznika použijeme jeho hexadecimálnu hodnotu %3F a podobne.

Samozrejme, že môžeme vyskúšať prístup z mobilného zariadenia.

Obrázok 4.10 - Prístup k údajom z mobilného zariadenia cez HTTP

Ak sa pozrieme na výsledok vyhľadávania, nie vždy sa hodí XML formát. Vtedy musíme napísať XSL šablónu v ktorej určíme formát výstupu údajov na HTML stránke klienta, alebo vytvori5 vhodn[aplika4n[logiku.

Niekedy nás zaujímajú len určitá podmnožina údajov, napríklad údaje o osobe s priezviskom Davolio., Samozrejme môžeme použiť URL adresu, ktorá bude obsahovať príkaz SELECT s klauzulou **WHERE** LastName='**Davolio**'.

http://llhome/vd_nwind?sql=SELECT+*+FROM+Employees+WHERE+LastName='Davolio'+FOR+XML+AUTO&root=ROOT

Ak si pozrieme naposledy zadanú URL adresu, ťažko môžeme požadovať od bežného používateľa, aby takýmto spôsobom zakaždým zisťoval údaje o konkrétnej osobe. Zjednodušenie prinesie volanie uloženej procedúry ParamVypisEmp, ktorú vytvoríme pomocou skriptu:

```
CREATE PROCEDURE ParamVypisEmp @Meno VARCHAR(10)
AS
    SELECT * FROM Employee
    WHERE LastName=@Meno
    FOR XML AUTO
GO
```

a šablóny **Procedura1.xml**:

```
<ROOT xmlns:sql='urn:schemas-microsoft-com:xml-sql'>
  <sql:header>
       <sql:param name='Meno'>Podmienka</sql:param>
  </sql:header>
  <sql:query >
      exec ParamVypisEmp @Meno
  </sql:query>
</ROOT>
```

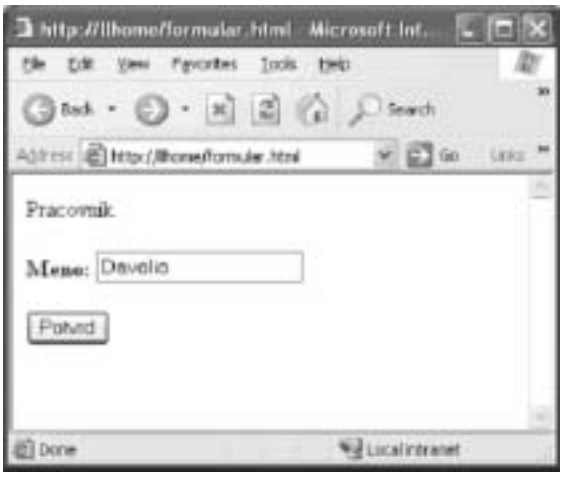
Vidíme že po týchto úpravách je výpis údajov o pracovníkovi oveľa jednoduchší:

http://llhome/vd_nwind/template/Procedura1.xml?Meno=Davolio

Takúto URL adresu však vygenerujeme jednoduchým formulárom na HTML stránke (formular.html):

```
Pracovnik
FORM ACTION="http://llhome/vd_nwind/template/Procedura1.xml" METHOD="GET" ><P>
<B>Meno:</B>
<TD><INPUT id=Meno name=Meno> </P><P>
<INPUT TYPE=submit value=Potvrd> </P>
</FORM>
```

Výsledná podoba HTML stránky pre vyhľadávanie údajov je potom jednoduchá a zrozumiteľná pre každého klienta.



Obrázok 4.11 - Prístup k údajom z mobilného zariadenia cez HTTP

SQL Server CE 2.0

Vo fyzickom slova zmysle SQL Server CE je realizovaný ako knižnica typu DLL, ktorá má približne jeden až jeden a pol megabajtu. Po technickej stránke nejde o službu na pozadí ako je to v prípade SQL Servera 2000, ale SQL Server CE je vlastne OLE DB provider, ktorý podporuje transakcie, viacnásobné indexy a referenčnú integritu. Na rozdiel od SQL Servera 2000 nie sú podporované pohľady, uložené procedúry, spúšte (triggery) ani procedurálny jazyk T-SQL. SQL Server CE 2.0 je navrhnutý tak, aby minimalizoval spotrebu systémových zdrojov, hlavne pamäťovej kapacity mobilného zariadenia. Webová stránka produktu je na adrese
http://www.microsoft.com/sql/ce/default.asp

Obrázok 4.12 – logo

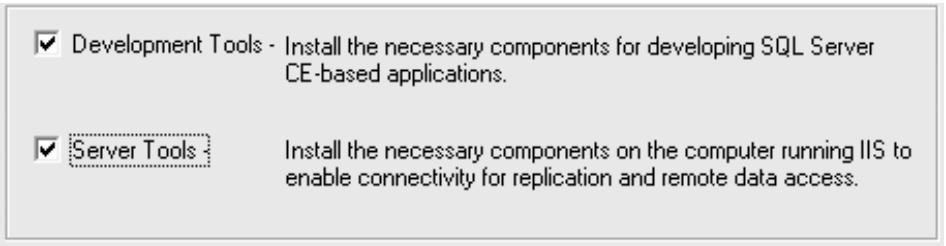
Priblížme si niektoré vlastnosti a parametre SQL Servera 2.0 aspoň v hrubých črtách
- Jazyk SQL kompatibilný s produktom SQL Server 2000
- efektívne algoritmy vyhľadávania údajov na princípe B Stromov
- široký rozsah dátových typov:
  - o TINYINT, SMALLINT, INTEGER, BIGINT
  - o REAL, NUMERIC, FLOAT
  - o BIT, BINARY, VARBINARY, IMAGE
  - o UNICODE znakové dátové typy NATIONAL CHARACTER, NATIONAL CHARACTER VARYING, NTEXT
  - o MONEY, DATETIME, UNIQUEIDENTIFIER
- 249 indexov pre tabuľku, viacstĺpcové indexy
- podpora NULL
- 128-bitové súborové šifrovanie pre databázový súbor (heslová ochrana)
- DDL: vytváranie a editovanie návrhovej štruktúry tabuliek, vytváranie databáz, referenčná integrita..
- DML: INSERT, UPDATE, DELETE
- SELECT: UNION, SET agregačné funkcie, INNER/OUTER JOIN, subselect, GROUP BY/HAVING
- rolovacie a dopredné kurzory
- veľkosť databázy nad 2GB
- BLOB objekty nad 1 GB

Inštalácia SQL Servera CE 2.0

Pred inštaláciou produktu SQL Server CE 2.0 potrebujeme mať nainštalované nasledovné softvérové komponenty:
– Vývojové prostredie (Microsoft Visual Studio.NET alebo Microsoft eMbedded Visual Tools)
– Microsoft SQL Server 2000
– Microsoft Internet Information Services (IIS)
– Microsoft Windows CE hardvér (prípadne emulátor)
– Microsoft ActiveSync® vo verzii 3.5 a vyššie (voliteľné)

Nová verzia vývojového prostredia Visual Studio.NET 2003 už obsahuje SQL CE 2.0 vrátane inštalačných súborov (v adresári
C:\Program Files\Microsoft Visual Studio .NET 2003\CompactFrameworkSDK\v1.0.5000\Windows CE)

V prvom inštalačnom dialógu volíme, či chceme nainštalovať len vývojárske, alebo aj serverové komponenty.

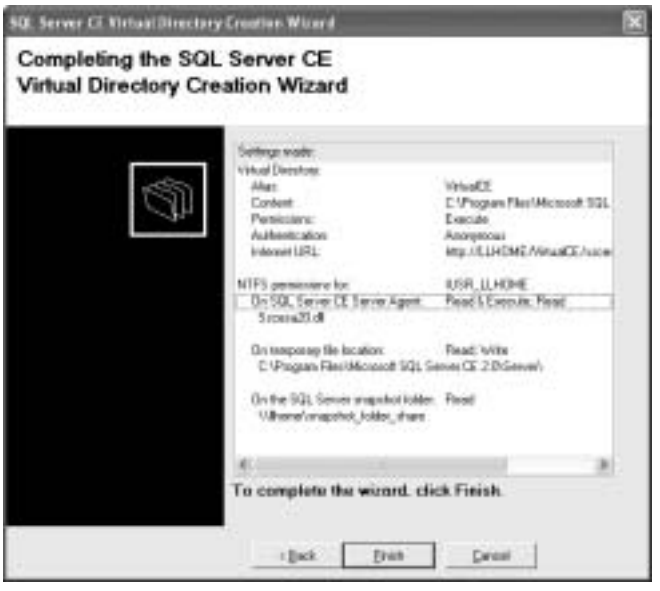


Obrázok 4.13 – inštalačný dialóg

SQL Server CE 2.0 a serverové nástroje budú implicitne nainštalované do adresárov:
C:\Program Files\Microsoft SQL Server CE 2.0
C:\Program Files\Microsoft SQL Server CE 2.0\Server

Po ukončení inštalácie môžeme v poslednom dialógu aktivovať sprievodcu vytvorením a konfiguráciou virtuálneho adresára.

Obrázok 4.14 – Sprievodca vytvorením virtuálneho adresára

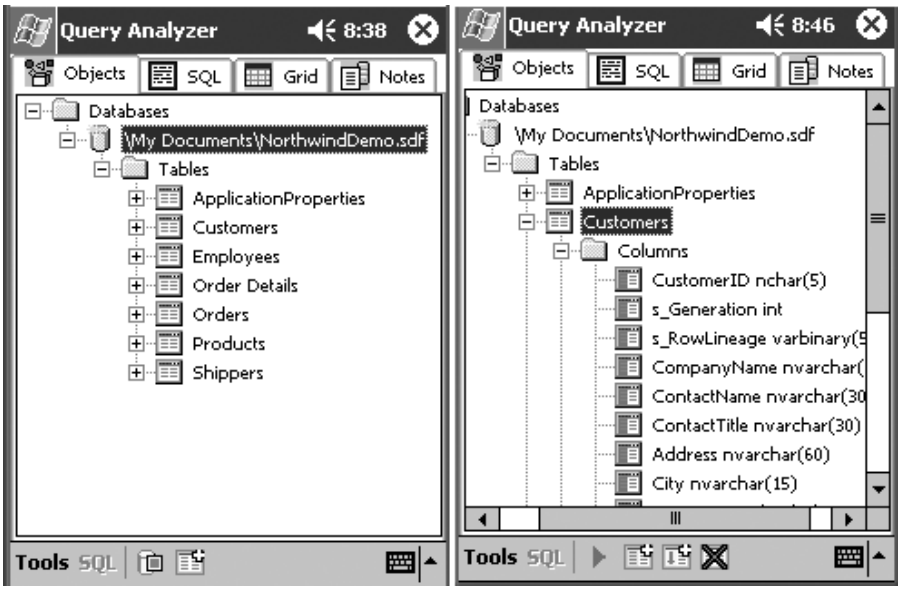


Obrázok 4.15 – Sprievodca vytvorením virtuálneho adresára

Virtuálny adresár môže obsahovať DLL-ku SQL Server Agent pre replikácie a RDA (Remote Data Acess) (súbor sscesa20.dll). V nasledujúcich dialógoch nastavíme prístupové parametre pre virtuálny adresár.

Konzolová aplikácia Query Analyzer

Samotný SQL Server CE je rozhranie medzi aplikáciou a databázou. Ak chceme administrovať databázu, pracovať s údajmi v databáze, napríklad generovať testovacie výpisy, môžeme použiť aplikáciu ISQLW20 (alebo ISQLWCE), ktorá je ekvivalentom aplikácie Query Analyser známej z SQL Servera 2000. Jeho inštalácia je jednoduchá. Stačí súbor **Isqlw20.exe** nakopírovať do adresára **Program Files** V prípade inštalácie databázovej aplikácie vyvinutej pomocou Visual Studia .NET 2003 je táto aplikácia nakopírovaná automaticky.

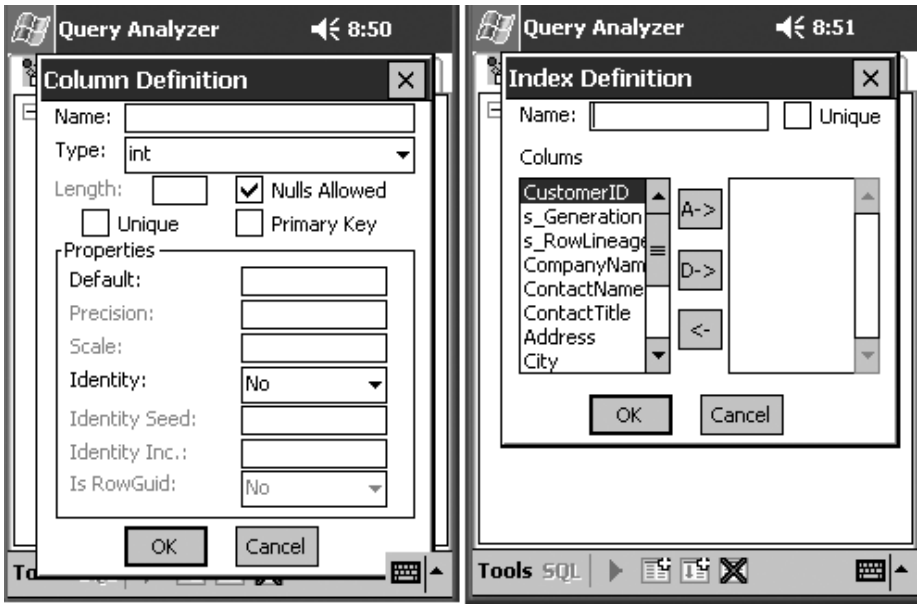


Obrázok 4.16 – Aplikácia Query analyse Záložka Objects

Hlavné okno aplikácie má štyri záložky
- Objects
- SQL
- Grid
- Notes

Objects

Po pripojení sa k databáze na mobilnom zariadení (súbor s príponou SDF) máme prístup k návrhovej štruktúre databázy, môžeme pridávať a editovat jednak na úrovni databázových tabuliek, indexov, prípadne stĺpcov



Obrázok 4.17 – Aplikácia Query analyser – dialógy pre vytvorenie nového stĺpca a indexu.

Zložky SQL, Grid a Notes

Tieto zložky slúžia pre zadávanie SQL príkazu (SQL), zobrazenie výsledkov dotazu (Grid) a oznamov databázového servera (Notes). Dohromady to veľmi nápadne pripomína aplikáciu Query Analyzer známu z SQL Servera 2000.

Obrázok 4.18 – Aplikácia Query analyser – dialógy pre vytvorenie nového stĺpca a indexu.

SQL SERVER CE 2.0 – parametre

Azda najlepším a najvýstižnejším spôsobom predstavenia databázového servera je zoznam kľúčových slov, dátových typov a zoznamu funkcií.

Zoznam kľúčových slov

@@IDENTITY
ADD
ALL
ALTER
AND
ANY
AS
ASC
AUTHORIZATION
AVG
BACKUP
BEGIN
BETWEEN
BREAK
BROWSE
BULK
BY
CASCADE
CASE
CHECK
CHECKPOINT
CLOSE
CLUSTERED
COALESCE
COLLATE
COLUMN
COMMIT
COMPUTE
CONSTRAINT
CONTAINS
CONTAINSTABLE
CONTINUE
CONVERT
COUNT
CREATE
CROSS
CURRENT
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
CURSOR
DATABASE
DATABASEPASSWORD
DATEADD
DATEDIFF
DATENAME
DATEPART
DBCC
DEALLOCATE
DECLARE
DEFAULT
DELETE
DENY
DESC
DISK
DISTINCT
DISTRIBUTED
DOUBLE
DROP
DUMP
ELSE
ENCRYPTION
END
ERRLVL
ESCAPE
EXCEPT
EXEC
EXECUTE
EXISTS
EXIT
EXPRESSION
FETCH
FILE
FILLFACTOR
FOR
FOREIGN
FREETEXT
FREETEXTTABLE
FROM
FULL
FUNCTION
GOTO
GRANT
GROUP
HAVING
HOLDLOCK
IDENTITY
IDENTITY_INSERT
IDENTITYCOL
IF
IN
INDEX
INNER
INSERT
INTERSECT
INTO
IS
JOIN
KEY
KILL
LEFT
LIKE
LINENO
LOAD
MAX
MIN
NATIONAL
NOCHECK
NONCLUSTERED
NOT
NULL
NULLIF
OF
OFF
OFFSETS
ON
OPEN
OPENDATASOURCE
OPENQUERY
OPENROWSET
OPENXML
OPTION
OR
ORDER
OUTER
OVER
PERCENT
PLAN
PRECISION
PRIMARY
PRINT
PROC
PROCEDURE
PUBLIC
RAISERROR
READ
READTEXT
RECONFIGURE
REFERENCES
REPLICATION
RESTORE
RESTRICT
RETURN
REVOKE
RIGHT
ROLLBACK
ROWCOUNT
ROWGUIDCOL
RULE
SAVE
SCHEMA
SELECT
SESSION_USER
SET
SETUSER
SHUTDOWN
SOME
STATISTICS
SUM
SYSTEM_USER
TABLE
TEXTSIZE
THEN
TO
TOP
TRAN
TRANSACTION
TRIGGER
TRUNCATE
TSEQUAL
UNION
UNIQUE
UPDATE
UPDATETEXT
USE
USER
VALUES
VARYING
VIEW
WAITFOR
WHEN
WHERE
WHILE
WITH
WRITETEXT

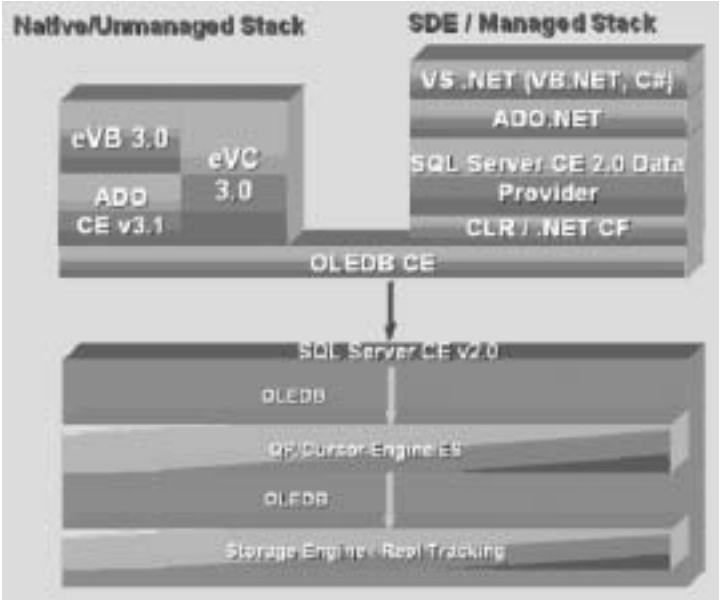
Tabuľka dátových typov

Dátový typ	Popis
bigint	Integer (celé číslo) v rozsahu od -2^63 (- 9,223,372,036,854,775,808) do 2^63-1 (9,223,372,036,854,775,807). Zaberie 8 bajtov.
integer	Integer (whole number) (celé číslo) v rozsahu od -2^31 (-2,147,483,648) do 2^31 - 1 (2,147,483,647).
smallint	Integer v rozsahu od -32,768 do 32,767. Zaberá 2 bajty.
tinyint	Integer v rozsahu od 0 do 255. Zaberá 1 bajt.
bit	Integer s hodnotami 0 alebo 1.
numeric *(p, s)*	Numerické údaje s pevnou rádovou čiarkou v rozsahu od -10^38 +1 do 10^38 -1. Parametrom p určujeme presnosť (počet platných číslic) od 1 do 38
money	Peňažné dáta v rozsahu od -2^63 (- 922,337,203,685,477.5808) do 2^63 - 1 (922,337,203,685,477.5807), Zaberie 8 bajtov..
float	Číslo s pohyblivou rádovou čiarkou v rozsahu od -1.79E + 308 do 1.79E + 308. Zaberie 8 bajtov.
real	Číslo s pohyblivou rádovou čiarkou v rozsahu od -3.40E + 38 do 3.40E + 38.
datetime	Dátum a čas v rozsahu od 1.1.1753, do 31.12. 9999, s presnosťou na tretinu stotiny sekundy (3.33 millisekúnd). Pozostáva z dvoch 4 bajtových integer čísiel. Prvé udáva počet dní od 1.1.1900 a druhé čas v rámci dňa v milisekundách od polnoci.
national character *(n)* Synonym: **nchar** *(n)*	Reťazec Unicode s pevnou dĺžkou max 255 znakov. Implicitná dĺžka = 1
national character varying *(n)* Synonym: **nvarchar** *(n)*	Reťazec Unicode s premenlivou dĺžkou max 255 znakov. Implicitná dĺžka = 1Storage size, in bytes, is two times the number of characters
ntext	Reťazec Unicode s premenlivou dĺžkou max (2^30 - 2) / 2 (536,870,911) znakov.
binary *(n)*	Binárne údaje s pevnou dĺžkou max 510 bajtov. Implicitná dĺžka = 1
varbinary *(n)*	Binárne údaje s premenlivou dĺžkou max 510 bajtov. Implicitná dĺžka = 1
image	binárne údaje s premenlivou dĺžkou maxf 2^30 - 1 (1,073,741,823) bajtov.
uniqueidentifier	Globálny unikátny identifikátor. Zaberá 16 bajtov.

Zoznam funkcií dostupných v jazyku SQL

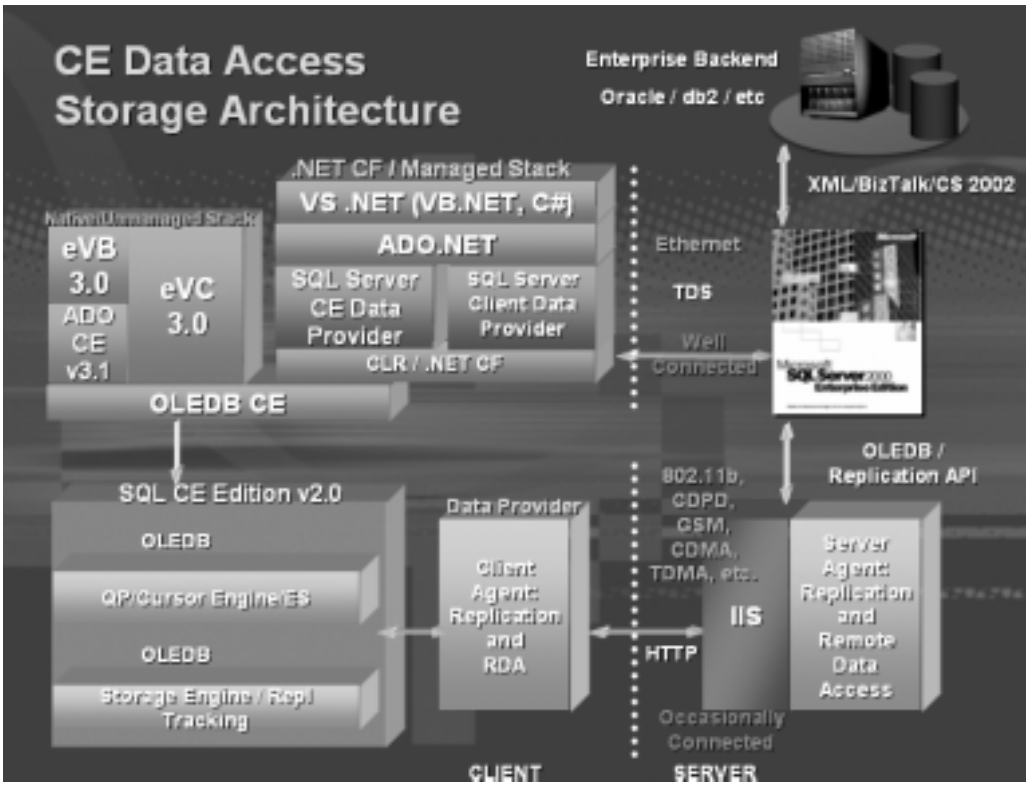
Funkcie	Popis
Agregačné funkcie **AVG, COUNT, MAX, MIN, SUM**	Na základe množiny údajov vypočítajú výslednú hodnotu, napríklad maximum, minimum, priemer... . S výnimkou funkcie COUNT agregačné funkcie ignorujú hodnoty NULL.
Funkcie pre prácu s dátumom a časom **GRTDATE, DATEADD DATEDIFF, DATENAME, DATEPART**	Pracujú s dátumom a časom. Vracajú dátumovú a časovú hodnotu, prípadne reťazcovú alebo numerickú hodnotu.
Matematické funkcie **ABS, ACOS, ASIN, ATAN, ATN2, CEILING, COS, COT, DEGREES, EXP, FLOOR, LOG, LOG10, PI, POWER, RADIANS, RAND, ROUND, SIGN, SIN, SQRT, TAN**	Na základe vstupných hodnôt vypočítajú numerický výsledok.
Reťazcové funkcie **NCHAR, CHARINDEX, LEN, LOWER, LTRIM, PATINDEX, REPLACE, REPLICATE, RTRIM, SPACE, STR, STUFF, SUBSTRING, UNICODE, UPPER**	Na základe reťazcových vstupných hodnôt vypočítajú reťazcový, alebo numerický výsledok.
Systémové funkcie **@@IDENTITY, CASE, COALESCE, CONVERT, DATALENGTH, NEWID**	Vykonávajú operácie a vracajú parametre nastavenia SQL CE.

Na obrázku vidíme možnosti prístupu aplikácií k údajom pod správou SQL Servera CE.



Obrázok 4.19 – SQL Server

Táto schéma je len určitou podmnožinou. Z globálneho hľadiska lepšie vystihuje situáciu schéma rozšírená o serverové okolie.



Obrázok 4.20 – Architektúra prístupu k údajom

Technológie SQL Servera CE 2.0 zapúzdrené v Compact Frameworku obsahujú dve zaujímavé komponenty. Jednak **System.Data.SqlClient** pre priamu prístup k údajom pod správou databázového servera SQL Server 2000 (alebo verzie 7.0). Pokiaľ využijeme tento spôsob prístupu, máme k dispozícii všetky vymoženosti poskytované „veľkým“ SQL Serverom, ako sú napríklad procedurálny jazyk T-SQL, uložené procedúry a podobne. Druhá komponenta **System.Data.SqlServerCE** je komponenta typu Local Data Provider. Kód tejto komponenty (beží samozrejme pod procesorom mobilného zariadenia) umožňuje aplikáciám využívať prístup k lokálnym údajom a plnú funkcionalitu SQL Servera CE 2.0. Vývojárom databázových aplikácií určite netreba pripomínať nevyhnutnosť ošetrenia chybových stavov a výnimiek. Na tieto účely je určený systém výnimiek zapúzdrený v **System.Data.SqlServerCe.SqlCeException**

Pri replikácii údajov medzi databázami musíme prihliadať na kompatibilitu dátových typov. Pomôže nám tabuľka:

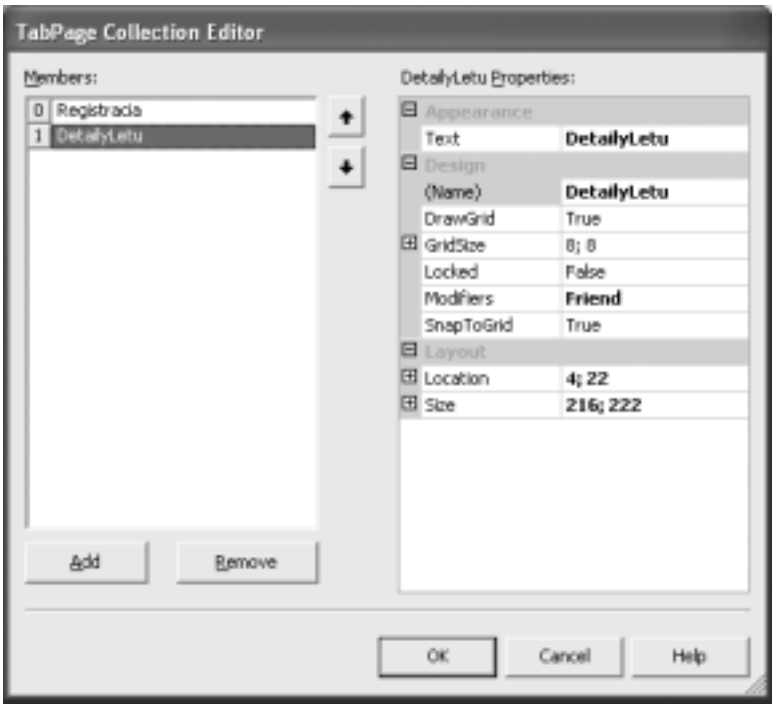
SQL Server 2000	**SQL Server CE**
BIGINT (INT 8)	BIGINT (INT 8)
BINARY (n)	BINARY (n) alebo IMAGE (nad 510 bajtov)
BIT	BIT
CHARACTER (synonym CHAR)	NATIONAL CHARACTER alebo NTEXT (nad 255 znakov)
CHARACTER VARYING (synonym VARCHAR)	NATIONAL CHARACTER VARYING alebo NTEXT (nad 255 znakov)
COMPUTED COLUMNS	–
DATETIMESMALLDATETIME	DATETIME
DOUBLE PRECISION	DOUBLE PRECISION
FLOAT	FLOAT
IMAGE	IMAGE
INTEGER (INT 4) synonym INT	INTEGER (INT 4)
MONEYSMALLMONEY	MONEY
NATIONAL CHARACTERsynonym NCHAR	NATIONAL CHARACTER alebo NTEXT (nad 255 znakov)
NTEXT	NTEXT
NUMERIC synonyma DECIMAL,DEC	NUMERIC
NATIONAL CHARACTER VARYING synonym NVARCHAR	NATIONAL CHARACTER VARYING alebo NTEXT (nad 255 znakov)
REAL	REAL
SMALLINT (INT 2)	SMALLINT (INT 2)
SQL_VARIANT	NTEXT
TEXT	NTEXT
TIMESTAMP	
TINYINT (INT 1)	TINYINT (INT 1)
UNIQUEIDENTIFIER	UNIQUEIDENTIFIER
VARBINARY (n)	VARBINARY(n) alebo IMAGE (nad 510 bajtov)

Zadanie úlohy číslo 4.1: Vytvorte databázovú aplikáciu, námetom ktorej bude agenda x-mílového klubu fiktívnych aerolinii.

Riešenie úlohy číslo 4.1: Vo Visual Studiu .NET vytvoríme nový projekt typu **Smart Device Application**. v jazyku **Visual Basic**. Projekt nazveme napríklad KlubSDA. V prvom kroku návrhu aplikačného formulára **Form1** umiestnime do neho komponentu typu **TabControl**.

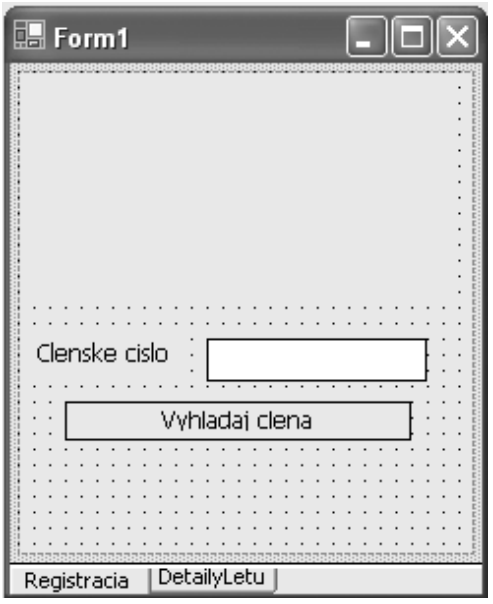
Obrázok 4.21 – TabControl Properties

V okne dialógu **TabPage Collection Editor** pridáme tlačidlom **Add** dve záložky. Pre prvú z nich nastavíme parametre **text** a **Name** na **Registracia** a pre druhu na **DetailyLetu**.



Obrázok 4.22 – Záložky TabControl

Do záložky Registrácie umiestnime komponenty **PictureBox, Label, TextBox** a **Button**. Ich rozmiestnenie na ploche aplikačného formulára najlepšie ukáže obrázok



Obrázok 4.23 – Návrhové zobrazenie záložky Registrácia

V okne Solution Explorer klikneme pravým tlačidlom myši na položku PrikladSDA (druhá zhora) vyberieme z menu položku **Add|Add Existing Item...** a pridáme do projektu obrázok BYA.BMP z jeho aktuálneho umiestnenia. Klikneme na túto komponentu a v okne Properties zmeníme parameter **Build Action** na **Content**

V ďalšom kroku pridáme kód pre aktiváciu komponenty pre vstup údajov – Soft Input Panel (SIP). V okne Solution Explorer klikneme pravým tlačidlom myši na položku Klub (druhá zhora) vyberieme z menu položku **Add|Add Existing Item...** a pridáme do projektu súbor **SipFunction.vb**. kde je funkcia pre aktiváciu panela pre zadávanie znakov

```
Public Class InputPanel
    <System.Runtime.InteropServices.DllImport("coredll.dll")> _
    Private Shared Function SipShowIM(ByVal dwFlag As Integer) As Integer
        ' Leave function empty - DLLImport attribute forwards calls to SipShowIM to
        ' SipShowIM in coredll.DLL.
    End Function
    Public Shared WriteOnly Property Visible() As Boolean
        Set(ByVal Value As Boolean)
            If Value = True Then
                SipShowIM(1)
            Else
                SipShowIM(0)
            End If
        End Set
    End Property
End Class
```

Do metódy `Sub New()` za komentár `'Add any initialization` pridáme hrubo vyznačený kód. Toto platí pre Visual Studio.NET 2003. V predchádzajúcej verzii Visual Studia s nainštalovaným SDE tento kód umiestnime do tela metódy **Form1_Load()**.

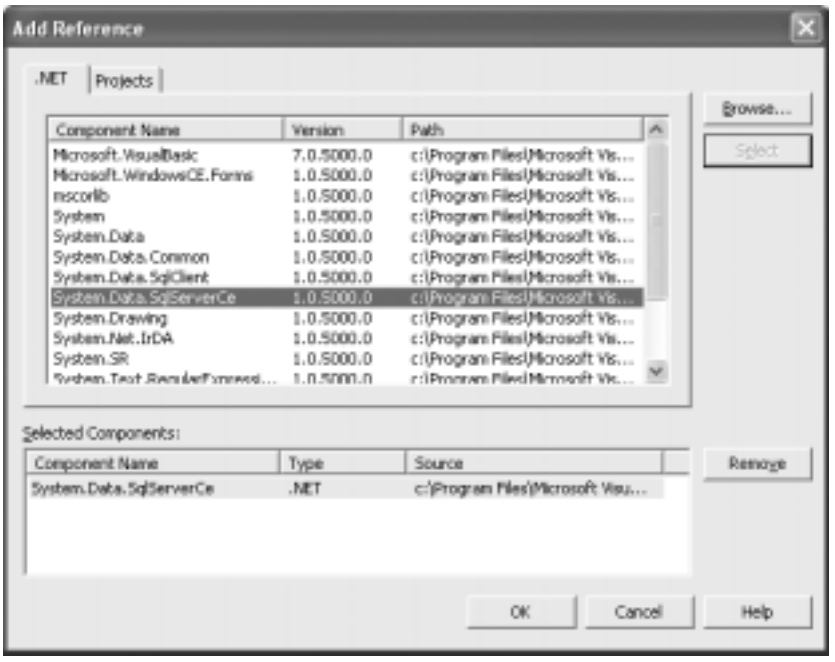
```
Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()
        'Add any initialization after the InitializeComponent() call

       'nastavime BYA.bmp ako 'content' a umiestnime ho do picture control
        Dim myImage = New Bitmap("/Program files/klubSDA/BYA.bmp")
        PictureBox1.Image = CType(myImage, Image)
        'skryjeme TabPages
        Me.TabControl1.TabPages.Item(1).Visible = False
        TextBox1.Text = "3"
        'SIP Control
        Dim oSIP As New InputPanel
        oSIP.Visible = True

    End Sub
```

Teraz môžeme aplikáciu zostaviť a vyskúšať na emulátore prípadne na reálnom mobilnom zariadení.

Obrázok 4.24 – Prvé spustenie v emulátore Pocket PC

V okne Solution explorer pomocou volby Add Reference pridáme referenciu na System.Data.SqlServerCe. V prípade, že táto referencia v zozname nie je, pridáme ju pomocou tlačidla Browse. Nájdeme súbor System.Data.SqlServerCe.DLL v adresári
C:\program files\Microsoft Visual Studio .NET 2003\CompactFrameworkSDK\v1.0.5000\Windows CE

V staršej verzii Visual Studia.NET je tento súbor s nazvom System.Data.SqlServerCe.DLL v adresári:
C:\program files\Microsoft Visual Studio .NET\CompactFrameworkSDK\v1.0.330\Windows CE



Obrázok 4.25 – Pridanie referencie na SQL Server CE

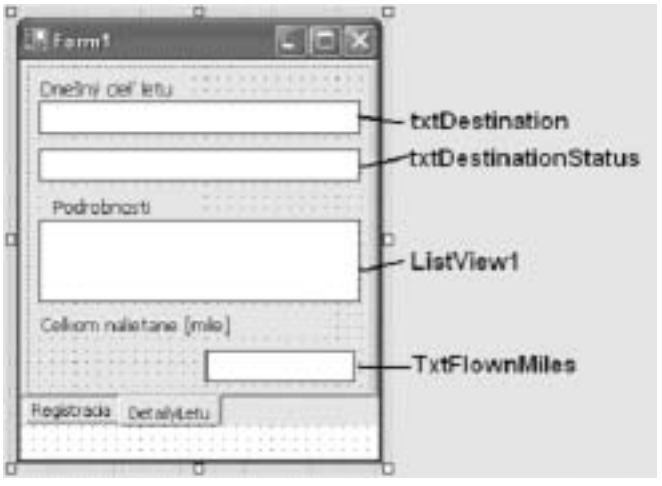
Do projektu pridáme podprogramy, na kódoch ktorých môžeme vidieť príklad managed kódu pre prácu s objektmi ADO.NET a kód pre replikáciu. Podprogram **CheckDB**() najskôr skontroluje, či predmetná databáza už existuje. K tomu využíva metódu **System.IO.File.Exists**. Ak predmetná databáza (súbor s príponou SDF) neexistuje, vytvorí sa pomocou metódy **CreateDatabase()**.

```
Sub CheckDB()
      Try
          If Not System.IO.File.Exists _
          ("\my documents\SQLCESDELab.sdf") Then
              'Vytvorenie inštancie objektu SQL Server CE Engine
              Dim en As New Engine _
              ("data source=\my documents\SQLCESDELab.sdf")
              'Vytvorenie prázdnej databázy
              en.CreateDatabase()
              Synch()
          End If
      Catch err As SqlCeException
          ErrorDisplay(err)
      Catch err As Exception
          MsgBox("There was an error: " & err.ToString())
      End Try
  End Sub
```

Obsah databáz je synchronizovaný volaním metódy **`Synchronize()`** v procedúre **`Synch()`**

```
    Sub Synch()
        Try
            ' Vytvorenie objektu pre replikáciu
            Dim cerepl As New Replication()
            With cerepl
                .Publisher = "llhome"
                .PublisherDatabase = "SQLCESDELab"
                .PublisherLogin = "sa"
                .PublisherPassword = "sqlce"
                .Publication = "SQLCESDELab"
                .Subscriber = "SQLCENETCFLab"
                .SubscriberConnectionString = _
                "Provider=Microsoft.SQLServer.OLEDB.CE.2.0;Data Source=\my
documents\SQLCESDELab.sdf"
                .InternetUrl = _
                "http://llhome/ssceweb2/sscesa20.dll"
                .Synchronize()
            End With
        Catch err As SqlCeException
            ErrorDisplay(err)
        End Try
    End Sub
```

Pokračujeme v návrhu druhej záložku aplikačného formulára. Okrem komponentov typu Label, doplníme komponenty **txtDestination**, **txtDestinationStatus** a **TxtFlownMiles** typu TextBox a komponentu ListView1.



Obrázok 4.25 – Návrh druhej záložky aplikačného formulára

doplníme procedúry **`Connect(),Disconnect()  a ErrorDisplay`**.

```
    Sub Connect()
        'ak databáza nie je prazdna
        Dim MyFile As New System.IO.FileInfo("\my documents\SQLCESDELab.sdf")
        If MyFile.Length = 45056 Then
            Synch()
        End If
        Try
            cn.Open()
        Catch err As SqlCeException
            ErrorDisplay(err)
        End Try
    End Sub
```

```
    Sub Disconnect()
        Try
            cn.Close()
        Catch err As SqlCeException
            ErrorDisplay(err)
        End Try
    End Sub
```

Všimnime si zostavenie reťazca,ktorý bude obsahovať prípadné chyby

```
    Sub ErrorDisplay(ByVal expSQL As SqlCeException)
        Dim errSQL As SqlCeError
        'vypis chyby do MsgBoxu
        For Each errSQL In expSQL.Errors
            MsgBox(errSQL.NativeError.ToString() & " " & errSQL.Source & " " &
errSQL.HResult.ToString() & " " & errSQL.Message)
        Next
    End Sub
```

Nasleduje obsluha tlačidla Buton1

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim strMembershipNumber As String = TextBox1.Text
        If (CheckMembershipDatabase(strMembershipNumber) = True) Then
            'zmena tabulky
            TabControl1.SelectedIndex = 1
            Me.TabControl1.TabPages.Item(1).Visible = True
        Else
            MsgBox("Invalid Member Number")
            Button1.Show()
        End If
        'SIP Control
        Dim oSIP As New InputPanel
        oSIP.Visible = False
    End Sub
    Function CheckMembershipDatabase(ByVal strMembershipNumber)
        Dim strDestination As String
        Dim lngMiles As Long
        Dim dtmDate As Date
        Dim strSQL As String
        Dim dr As SqlCeDataReader
        'pripojenie na lokálnu databázu
        Connect()

        'SQL dotaz
        strSQL = "SELECT M.MemberId, MemberName, Destination, FlightStatus, ArrivalDate,
FlownMiles FROM MemberShipData M INNER JOIN FlightData F ON F.MemberId = M.MemberId AND
M.MemberId = "
        strSQL &= strMembershipNumber & ";"

        Dim cmd As New SqlServerCe.SqlCeCommand(strSQL, cn)
        Try
            dr = cmd.ExecuteReader()
        Catch err As SqlCeException
            ErrorDisplay(err)
        Catch err As Exception
            MsgBox("There was an error: " & err.ToString())
```

```
End Try
If dr.Read = True Then
     'napln list view udajmi z databazy
     CheckMembershipDatabase = True
     Me.ListView1.Clear()
     ListView1.View = View.Details
     ListView1.GridLines = True
     'nastav stĺpce
     ListView1.Columns.Add("Date      ", -2, HorizontalAlignment.Left)
     ListView1.Columns.Add("Destination", -2, HorizontalAlignment.Left)
     ListView1.Columns.Add("Miles     ", -2, HorizontalAlignment.Left)

     'Napln komponenty typu TextBox
     txtDestination.Text = dr.Item("Destination")
     txtDestinationStatus.Text = dr.Item("FlightStatus")

     Dim iFlownMiles As Integer = dr.Item("FlownMiles")
     strDestination = dr.Item("Destination")
     lngMiles = dr.Item("FlownMiles")
     dtmDate = dr.Item("ArrivalDate")

     Dim item1 As New ListViewItem(dtmDate)
     Dim subItem1 As New ListViewItem.ListViewSubItem
     Dim subItem2 As New ListViewItem.ListViewSubItem

     subItem1.Text = strDestination
     subItem2.Text = lngMiles

     item1.SubItems.Add(subItem1)
     item1.SubItems.Add(subItem2)
     ListView1.Items.Add(item1)

     item1 = Nothing
     subItem1 = Nothing
     subItem2 = Nothing

     While dr.Read()
          iFlownMiles += dr.Item("FlownMiles")

          strDestination = dr.Item("Destination")
          lngMiles = dr.Item("FlownMiles")
          dtmDate = dr.Item("ArrivalDate")

          item1 = New ListViewItem(dtmDate)
          subItem1 = New ListViewItem.ListViewSubItem
          subItem2 = New ListViewItem.ListViewSubItem

          subItem1.Text = strDestination
          subItem2.Text = lngMiles

          item1.SubItems.Add(subItem1)
          item1.SubItems.Add(subItem2)
          ListView1.Items.Add(item1)

          item1 = Nothing
          subItem1 = Nothing
          subItem2 = Nothing
     End While
```

```
            txtFlownMiles.Text = System.Convert.ToString(iFlownMiles)
        Else
            'ak v databáze nie sú žiadne záznamy
            CheckMembershipDatabase = False
        End If
        dr.Close()
        Disconnect()
    End Function
```

Po úspešnom vytvorení a odladení kódu aplikácie môžeme túto spustiť na mobilnom zariadení. O aktuálnom obsahu databázy sa môžeme presvedčiť pomocou utility **SQLCE Query**.

Kapitola 5

Microsoft Mobile Internet Toolkit alebo po novom ASP.NET Mobile Controls

Prvá časť názvu tejto kapitoly je síce v dobe písania zborníka ešte aktuálna, dlho už však nebude. Microsoft Mobile Internet Toolkit bude nahradený technológiou **ASP.NET Mobile Controls**, ktorá bude priamo implementovaná do .NET Frameworku 1.1 a nového **Visual Studia .NET 2003** (kódový názov beta verzie je Everett). Predmetom kapitoly budú webové aplikácie, vytvorené pre mobilné zariadenia. Tieto aplikácie bežia na serveri, mobilné zariadenie sa teda využíva ako tenký klient. Pre ich vývoj využijeme možnosti technológie ASP.NET stránok. Rozsah použitia takto vyvinutých aplikácií je veľmi široký a pokrýva mobilné platformy od Pocket PC až po bežný mobilný telefón, ktorý disponuje protokolom WAP.

WAP – internet v mobilnom telefóne

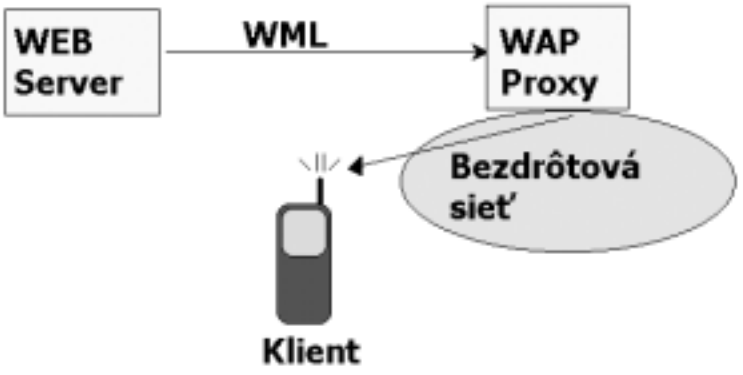
Keďže WAP je v súčasnosti dosť rozšírený, aj keď predstavy jeho tvorcov sa zrejme úplne nenaplnili, nezaškodí pár slov o tejto platforme. História protokolu WAP (Wireless Application Protocol) siaha len do roku 1997. Vtedy, 26 júna najvýznamnejší výrobcovia mobilných telefónov založili Fórum pre WAP. Definícia protokolu popisuje dátovú komunikáciu medzi mobilným zariadením a sieťou mobilného operátora. Prvá verzia WAP 1.0 bola zavedená do praxe v marci roku 1998. V decembri roku 1999 bol zverejnený predpis pre verziu 1.2. Hlavným prínosom tejto verzie bolo lepšie zabezpečenie, hlavne kôli elektronickému obchodu a bankovníctvu. Fórum pre bezdrôtový aplikačný protokol WAP je otvorená organizácia, ktorá združuje nielen výrobcov mobilných telefónov, ale aj operátorov mobilných sietí GSM, telekomunikačné spoločnosti, banky, elektronické obchody, ale aj vývojárov webových aplikácií, hier a podobne. Cieľom združenia je vytvárať a udržiavať normy pre prenos informácií prostredníctvom mobilných telefónov a iných bezdrôtových komunikačných zariadení, napríklad pagerov, personálnych digitálnych asistentov (PDA) a podobne. Nové možnosti týchto prenosných zariadení budú využité hlavne v elektronickom obchodovaní, homebankingu a prístupe na Internet.

Návrh protokolu pre bezdrôtové zariadenia si kladie za cieľ nezávislosť od výrobcov zariadení, prenosovej cesty a technologickej platformy serverov. Protokol je vhodný hlavne na prenos dát, ale aj hlasu, obrázkov ba dokonca aj na prenos komprimovaného videa.

Architektúra siete s protokolom WAP

Zjednodušene si môžeme architektúru bezdrôtovej siete s protokolom WAP premietnuť do troch vrstiev.

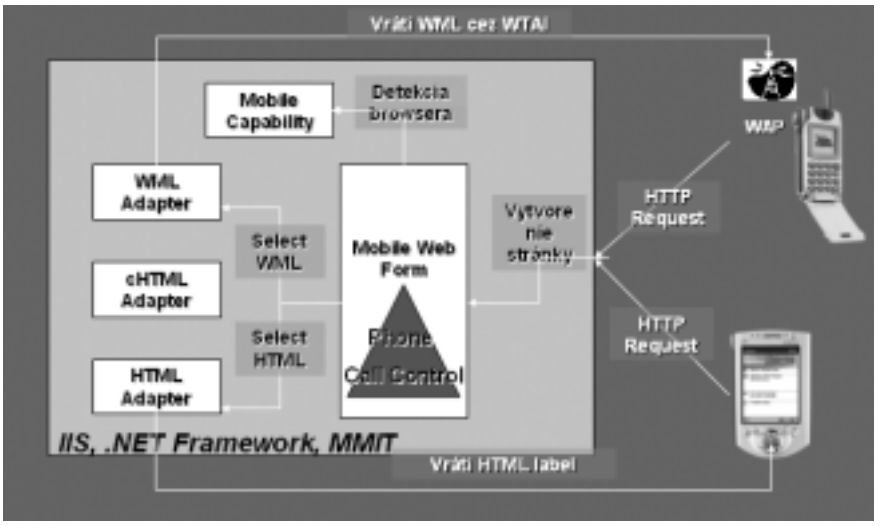
- **Vrstva klienta**
- **Prenosové médium**
- **Vrstva servera**



Vrstvu servera si môžeme zjednodušene popísať pomocou dvoch blokov.

Web server pracuje podobne ako u klasického internetu. Zo sprostredkujúceho serveru dostáva žiadosti v tvare URL a poskytuje odpovede v HTML

Sprostredkujúci systém si môžeme predstaviť ako Proxy server klasického Internetu. To znamená že voči WEB serveru vystupuje ako klient, ale tvorí server pre mobilného klienta, ktorý je pripojený prostredníctvom WAP protokolu.



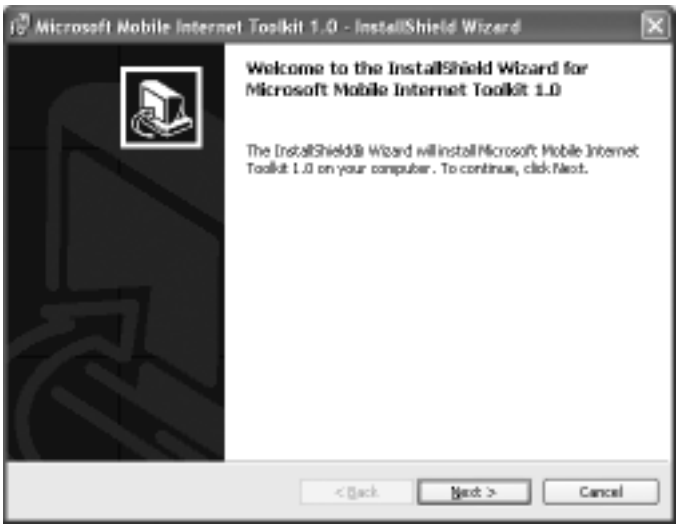
Microsoft Mobile Internet Toolkit (ASP .NET Mobile Controls),

Microsoft Mobile Internet Toolkit umožňuje vývoj takzvaných „tenkých“ aplikácií, ktorých aplikačná logika beží na serveri. Z mobilného zariadenia sa k nej pristupuje spravidla pomocou webového prehliadača. MMIT je založený na technológii ASP.NET stránok, HTML, cHTML (Compact HTML – ázijská špecialita) a WML (Wireless Markup Language). Bol testovaný na viac než osemdesiatich typoch mobilných zariadení. Uvedieme aspoň reprezentatívny prehľad zariadení dostupných v našich končinách:

- **ACCESS Compact:** NetFront 2.0, Fujitsu F503i, Mitsubishi D502i, Mitsubishi D503i, NEC N210i, NEC N502i, Sony SO503i
- **Ericsson 2.0:** Ericsson R380, Ericsson R320, Ericsson R520m, Ericsson T20s
- GoAmerica Go.Web: Compaq iPAQ H3650, Palm Vx, RIM Blackberry 857, RIM Blackberry 950, RIM Blackberry 957,
- **Microsoft Mobile Explorer:** Sony CMD-Z5, Sony CMD-J5, Benefon Q,
- **Microsoft Pocket Internet Explorer:** Casio Cassiopeia E-125, Compaq iPAQ H3630, Compaq iPAQ H3650, HP Jornada 720, Compaq iPAQ H3670
- **Nokia:** Nokia 3330, Nokia 6210, Nokia 7110, Nokia 9110i
- **Openwave UP.Browser 3.x:** Audiovox CDM-9000, Ericsson R280LX, Hitachi C407H, Kyocera QCP 2035A, Kyocera QCP 3035, LG V111, Mitsubishi T250, Motorola StarTAC 7868W, Motorola TimePort P8767, Samsung SCH-6100, Samsung SCH-850, Samsung SCH-8500, Samsung UpRoar M100, Sanyo C401SA, Sanyo SCP-4500, Sanyo SCP-5000, Sprint Touchpoint, Sprint Touchpoint 2200, Sprint Touchpoint 3000
- **Openwave UP.Browser 4.x:** Alcatel One Touch 701, Audiovox CDM-135, Audiovox CDM-9100, Motorola i1000plus, Motorola i2000plus, Motorola i50sx, Motorola i85s, Motorola T2288, Motorola TimePort P7382i, Motorola TimePort P7389, Motorola V100, Motorola V120c, Motorola V2288, Motorola V60c, Siemens C35i, Siemens S35i, Siemens SL45
- **Ostatné prehliadače:** Handspring Visor Platinum (Qualcomm Eudora Internet Suite 2.1; Blazer 1.0 and Omnisky 2.1.0.15), IBM WorkPad c505 (ilinx Xiino 1.01J), Kyocera QCP 6035 (Qualcomm Eudora 2.0), Nokia 9210 (Symbian Crystal 6.0), Palm VII (MyPalm 1.0), Palm Vx (AU-Systems 2.12181.1 and Omnisky 2.0.04), Palm m505 (MyPalm 1.1), Panasonic P210i, Panasonic P502i, Sharp J-SH04 (Original Equipment Manufacturer's Version 3.0), Sharp Zaurus MI-E1 (Original Equipment Manufacturer's Version 6.1), Sony CLIE PEG-N700C (ilinz Palmscape 4.0SJ), Toshiba J-T05 (Original Equipment Manufacturer's Version 3.0),

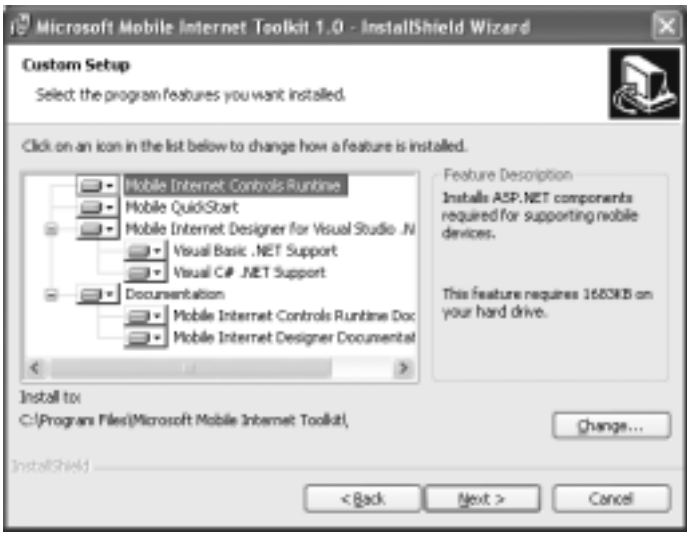
K dispozícii je už tzv. **Device Update 1**, kde je podporovaných viac než 130 zariadení (t.j 50 nových). V najbližšej dobe sa predpokladá dostupnosť **Device Update 2**.

MMIT je možné stiahnuť z webu Microsoftu, konkrétne zo sekcie Download z adresy
http://download.microsoft.com/download/VisualStudioNET/Install/RC/NT45XP/EN-US/MobileIT.exe
Netreba mať ani prílišné obavy z jeho sťahovania, nakoľko súbor má len niečo vyše 4MB.



Obrázok 5.1 - Inštalácia MMIT

V nasledujúcom dialógu sa môžeme rozhodnúť pre implicitnú inštaláciu, kedy sa nainštalujú všetky súčasti produktu, prípadne pre voliteľnú inštaláciu.

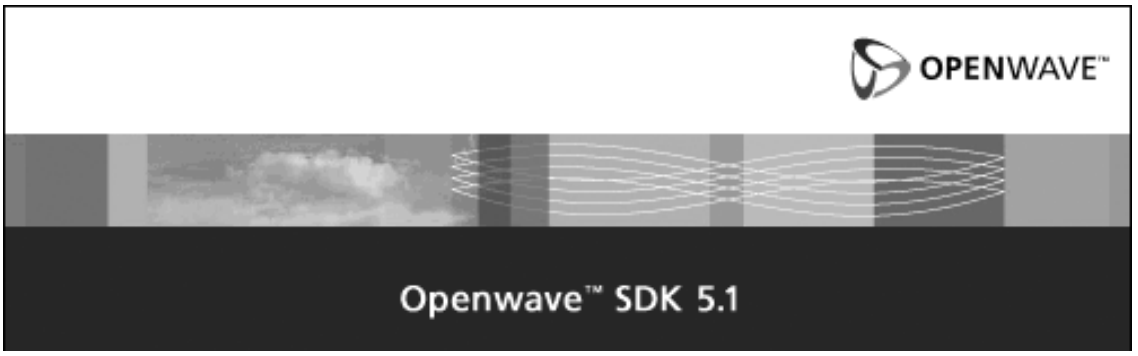


Obrázok 5.2 – Voliteľná inštalácia MMIT

Pre vývoj a ladenie aplikácií pre platformu Pocket PC budeme ešte potrebovať pravdepodobne **emulátor Pocket PC 2002** (rovnaký ako je dodávaný v eMVT). Tento emulátor môžeme voľne stiahnuť z webovej adresy *http://www.microsoft.com/mobile/downloads/emvt30.asp* alebo z inej webovej adresy *http://support.microsoft.com/directory/article.asp?id=kb;en-us;Q296904*

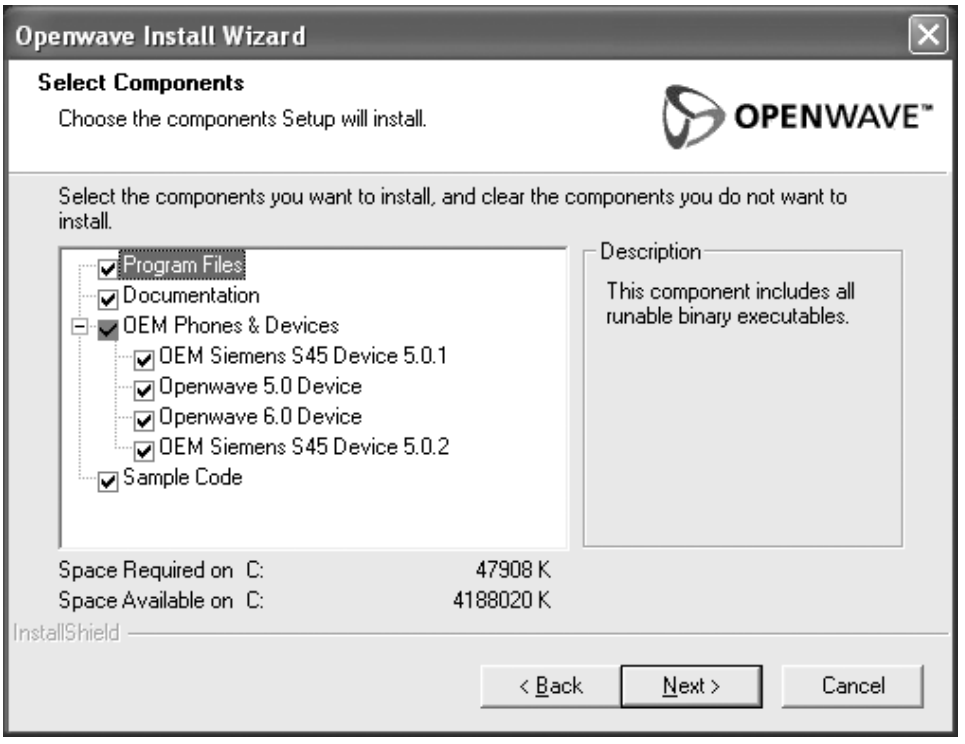
Pre ladenie WAP aplikácie, tu pôjde hlavne o uspôsobenie aplikácie zobrazovacím možnostiam displeja. Aj tu sú veľké rozdiely medzi telefónmi s pomerne veľkými displejmi, napríklad Ericsson R380, prípadne Motorola Accompli, a na druhej strane pomerne skromnými zobrazovacími možnosťami iných modelov, napríklad Ericsson T20. Aj toto usporiadanie aplikačnej logiky si môžeme otestovať na emltore. Existuje niekoľko „open“ emulátorov, napríklad Openwave SDK, alebo Microsoft Mobile Explorer Emulátor.

Openwave Emulátor.
Tento populárny a rozšírený emulátor môžeme získať z webu z adresy www.openwave.com



Obrázok 5.3 – Openwave SDK.5.1

Pri inštalácii si môžeme vybrať, aké typy zariadení chceme emulovať.



Obrázok 5.4 – Openwave SDK.5.1

Typ emulovaného zariadenia môžeme prepínať kedykoľvek počas behu SDK. V dialógu Select Device môžeme pre jednotlivé zariadenia zistiť podporované značkovacie jazyky, formáty obrázkov, znakovú sadu, veľkosť displeja a podobne.

Obrázok 5.5 – Openwave SDK.5.1 dialóg Select Device

Pre naše príklady budeme používať Openwave SDK vo verzii 5.1 len vo funkcii simulátora. Pre vlastný vývoj aplikácie budeme používať Visual Studio .NET s nainštalovaným Microsoft Mobile Internet Toolkitom

Obrázok 5.6 – Openwave SDK.5.1

Microsoft Mobile Explorer Emulátor

Tento prehliadač taktiež emuluje zobrazovacie možnosti rôznych mobilných zariadení. Pretože pri použití tohoto emulátora sú dalo by sa povedať „všetky kone z jednej stajne“, teda od Microsoftu a s nastavením Internet Information Servera si nemusíme robiť starosti, môžeme tento emulátor poradiť hlavne začiatočníkom, ktorí s administráciou webových a proxy serverov veľké skúsenosti nemajú. Získať **Microsoft Mobile Explorer Emulátor** je veľmi ľahké, môžeme ho voľne stiahnuť z webovej adresy
http://www.microsoft.com/mobile/phones/mme/mmemulator.asp alebo z webovej adresy
http://download.microsoft.com/download/VisualStudioNET/Install/3.0/NT45XP/EN-US/MME30.exe

Microsoft Mobile Explorer Emulátor môžeme aktivovať jednak vo vývojovom prostredí pomocou menu **View | Mobile Explorer Browser** alebo ako samostatný produkt z menu Windows. Môžeme si vybrať tri typy zariadení Large, Small a XP (na obr zľava doprava)

Obrázok 5.7 – Režimy Large, Small a XP

Prvá aplikácia vytvorená v MMIT

Nadpis odstavca trochu predbieha udalosti. Určite by bolo jednoduché spustiť Visual Sudio .NET a vytvoriť novú aplikáciu typu Mobile Web Application... ale keďže MMIT využíva technológiu ASP.NET, nezaškodí sa s touto technológiou zoznámiť trochu podrobnejšie. Zistíme, že prvú aplikáciu dokážeme pomerne hravo napísať aj v ľubovoľnom textovom editore.

ASP.NET

Hlavný rozdiel medzi technológiou ASP a ASP.NET je ten, že kódy na stránkach ASP.NET sú kompilované. Tým sa odstránila nutnosť analýzy a interpretácie jednotlivých riadkov pri každom prístupe klienta. Vznikne kompilovaný kód, ktorý je samozrejme oveľa rýchlejší. Pre ASP.NET stránky je vyčlenená súborová prípona **ASPX** (ASP stránky mali príponu súborov ASP). Stránky jednak zapúzdrujú vizuálne elementy pre komunikáciu s používateľom ktoré generujú príslušné udalosti, napríklad reakciu na kliknutie na tlačidlo a podobne. Tým je dosiahnuté nelineárne, udalosťami riadené spracovávanie kódu.

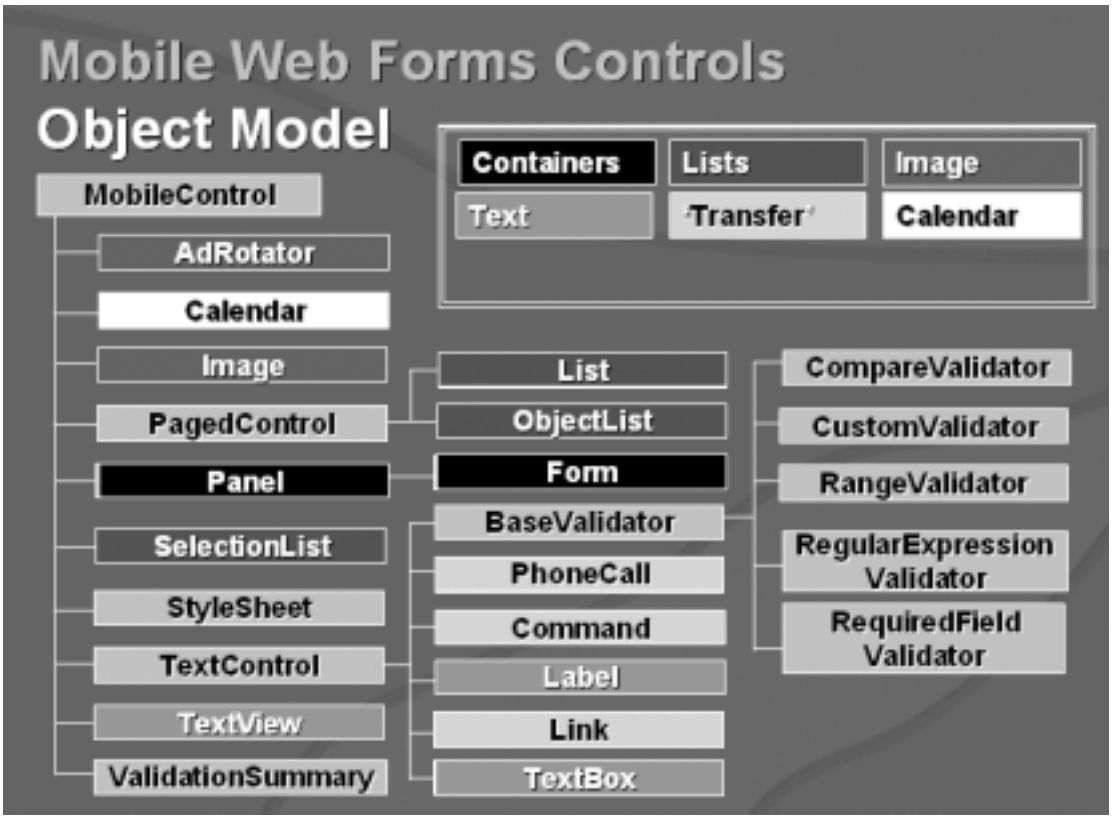
Pre umiestnenie našich vlastných ASP.NET môžeme využiť podobne ako u ASP stránok adresár
c:\inetpub\wwwroot
prípadne podadresáre. Potom budeme aplikácie volať prostredníctvom URL adresy http://localhost
Najjednoduchšia ASP.NET stránka, ktorá vypíše do okna prehliadača jednoduchý textový reťazec je

```
<Form runat="server">
      <asp:Label runat="server">
          Hello, World
       </asp:Label>
</Form>
```

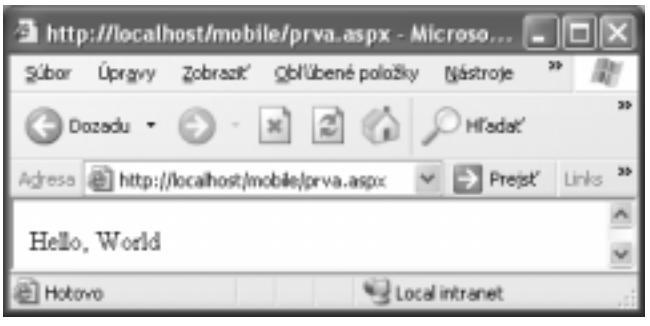
takúto ASP.NET stránku môžeme prepísať pre mobilné zariadenia nasledovne.

```
<%@ Page Inherits="MobilePage ..>
<%@ Register TagPrefix="mobile" ..>
<mobile:Form runat="server" ID="Form1">
      <mobile:Label runat="server">
            Hello, Mobile World
      </mobile:Label>
</mobile:Form>
```

Výber komponentov pre tvorbu formulárov aplikácií pre mobilné zariadenia nám priblíži schéma:



Podmienkou pre prácu s ASP.NET stránkami na serveri, alebo aj na lokálnom počítači je nainštalovaný **Internet Information Server** (IIS – súčasť Windows) a **.NET Framework**. Ak náš súbor nazveme prva.asxp a umiestnime ho do nejakého adresára, ktorý je dostupný pre Internet Information Server, napríklad ***c:\inetpub\wwwroot\mobile\prva.aspx*** potom môžeme zadať v okne prehliadača adresu *http://localhost/mobile/prva.aspx*



Obrázok 5.8 – Klasická ASP.NET aplikácia

Takáto stránka sa zobrazí aj prostredníctvom emulátora mobilného prehliadača

Obrázok 5.9 – ASP.NET aplikácia v okne Openwave simulátora

Aby sme v tom mali lepší prehľad, zosumarizujeme si, čo musíme mať nainštalované pre vývoj webových aplikácií pre mobilné zariadenia. Hrubým fontom sú označené komponenty, ktoré musíme mať nainštalované nutne.

- **Internet Information Services (IIS) a .NET Framework (obsahuje ASP.NET) server.**
- Visual Studio .NET
- **Microsoft Mobile Internet Toolkit**
- **Emulátor príslušnej mobilnej platformy**
- pre vývoj databázových aplikácií databázový server **SQL Server 2000**

Ako programovací jazyk môžeme opäť zvoliť Visual Basic a Visual C#. Rozdiely ukážeme na jednoduchom príklade.

Príklad v jazyku Visual Basic

(súbor prvaVB.aspx)

```
<%@ Page Language="VisualBasic" %>
<%
 'deklaracia premennych
  Dim nX As Integer
  Dim sNapis As String

 'priradenie hodnoty premennych
  sNapis = "Hello World VB"
  nX = 1

 'Vypis
  Response.Write(sNapis)
  Response.Write(nX)
%>
```

Príklad v jazyku C#

(súbor prvaCis.aspx)

```
%@ Page Language="C#" %>
<%
// deklaracia premennych
  int nX;
  String sNapis;

//priradenie hodnoty premennych
  sNapis = "Hello World C#";
  nX = 1;

//Vypis
  Response.Write(sNapis);
  Response.Write(nX);
%>
```

Aj tieto stránky sa zobrazia v okne emulátora. Pre zobrazenie v režime WAP ich musíme podobne ako našu prvú stránku doplniť príslušnými direktívami.

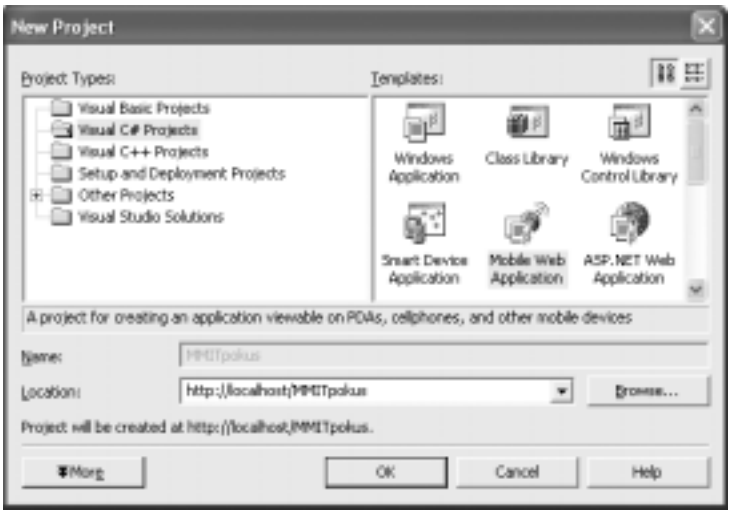
Obrázok 5.10 – ASP.NET aplikácie Visual Basic a C# v okne emulátora mobilného prehliadača

V ďalšom jednoduchom príklade sa pohráme s porovnaním zobrazovacích možností jednotlivých typov zariadení.

```
<%@ Page Language="C#" %>
<html>
  <body>
    <h3><font face="Verdana">Prva ASP.NET stranka</font></h3>
    <hr>
    <asp:label id="Message1" font-size="16" font-bold="true" forecolor="red"
                runat=server>Zdravime vyvojarov</asp:label><br>
    <asp:label id="Message2" font-size="20" font-italic="true" forecolor="blue"
                runat=server>Zdravime vyvojarov</asp:label><br>
    <asp:label id="Message3" font-size="24" font-underline="true" forecolor="green"
                runat=server>Zdravime vyvojarov</asp:label>
    </body>
</html>
```

Obrázok 5.11 -vzhľad ASP.NET aplikácie v rôznych prehliadačoch

V pokusoch s technológiou ASP.NET by sme mohli pokračovať, napríklad formulármi a podobne, no tu nám už výrazne pomôže Visual Studio.NET s nainštalovaným MMIT, alebo nové vývojové prostredie Visual Studio .NET 2003. Opäť máme na výber dva programovacie jazyky **Visual Basic** a **Visual C#**. Kliknutím na ikonu v príslušnej zložke môžeme vytvoriť novú aplikáciu typu **Mobile Web Application.**

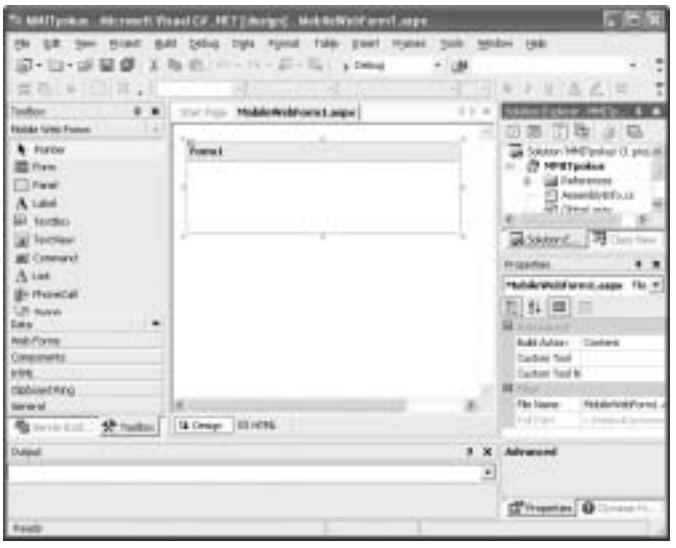


Obrázok 5.12 – Nový projekt – aplikácia typu Mobile Web Application

Projekt nejako pomenujeme, napríklad MMITpokus. Novo založený projekt bude potom k dispozícii pre vývoj, ladenie a testovanie na lokálnej adrese http://localhost/MMITPokus
Po zatlačení tlačidla OK sa stane niekoľko vecí. Globálne by sa to dalo povedať veľmi jednoducho: bol založený nový projekt. Aby sme sa v novom projekte, teda v súboroch, z ktorých sa skladá aj trocha vyznali a dokázali tak tento projekt zazálohovať, alebo prekopírovať na nejaké médium, potrebujeme „vypátrať“, kde sa vlastne náš projekt fyzicky nachádza. Odpoveď na túto otázku je veľmi jednoduchá a je vlastne len jedno miesto, kde by sa fungujúci webový projekt mohol nachádzať. Predsa v niektorom adresári pre publikovanie webových dokumentov, v našom prípade v adresári *C:\Inetpub\wwwroot* vznikol nový podadresár s názvom zhodným s názvom náš projektu, teda *MMITpokus*. V podadresári projektu je niekoľko súborov:

```
AssemblyInfo.cs                    Serial6.vsdisco
Global.asax                        Web.config
Global.asax.cs                     MobileWebForm1.aspx
Global.asax.resx                   MobileWebForm1.aspx.cs
MMITpokus.csproj                   MobileWebForm1.aspx.resx
MMITpokus.csproj.webinfo
```

Nás bude zaujímať z celého projektu zatiaľ hlavne súbor **MobileWebForm1.aspx**, teda ASP.NET stránka. Najskôr sa zoznámme so základným rozložením obrazovky Visual Studia .NET v režime MMIT. V ľavej časti máme Toolbox s ponukou komponentov, v strednej časti je pracovné okno, v ktorom môžeme mať zobrazený buď zdrojový kód, alebo návrhové zobrazenie a v pravej časti vidíme dva okná: **Solution Explorer** a **Properties.**



Obrázok 5.13 – Visual Studio .NET – MMIT projekt

Režim zobrazenia v strednej časti pracovnej plochy Visual Studia.NET nastavujeme pomocou záložiek **Design** a **HTML** v jeho spodnej časti. V režime Design máme zobrazený (zatiaľ) prázdny formulár, ktorý budeme neskôr dotvárať a v režime HTML môžeme písať prípadne editovať zdrojový kód ASP.NET stránky. Nezačíname z nuly, sprievodca vytvorením aplikácie (Application Wizard) už v etape vytvárania novej aplikácie vygeneroval v súbore **MobileWebForm1.aspx** kód:

```
<%@ Page language="c#" Codebehind="MobileWebForm1.aspx.cs"
Inherits="MMITpokus.MobileWebForm1" AutoEventWireup="false" %>
<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile, Version=1.0.3300.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a" %>

<meta name="GENERATOR" content="Microsoft Visual Studio 7.0">
<meta name="CODE_LANGUAGE" content="C#">
<meta name="vs_targetSchema" content="http://schemas.microsoft.com/Mobile/Page">
<body Xmlns:mobile="http://schemas.microsoft.com/Mobile/WebForm">
   <mobile:Form id="Form1" runat="server">

       </mobile:Form>
</body>
```

Ak by sme v tejto etape projektu neodolali a chceli by sme zobraziť novovytvorenú stránku (v našom prípade na adrese *http://localhost/MMITpokus/MobileWebForm1.aspx*) pomocou webového prehliadača, k úspechu by to spočiatku neviedlo

Server Error in '/MMITpokus' Application.

Parser Error

Description: An error occurred during the parsing of a resource required to service this request. Please review the following specific parse error details and modify your source file appropriately.

Parser Error Message: Could not load type 'MMITpokus.Global'.
Source Error:

```
Line 1:  <%@ Application Codebehind="Global.asax.cs" Inherits="MMITpokus.Global" %>
```

Source File: c:\inetpub\wwwroot\MMITpokus\global.asax **Line:** 1

Version Information: Microsoft .NET Framework Version:1.0.3705.0; ASP.NET Version:1.0.3705.0

Zostavenie a kompilácia aplikácie

Príčina neúspechu je v tomto prípade jasná. Projekt totiž obsahuje aj kód v jazyku C#, ktorý je v súbore **MobileWebForm1.aspx.cs** a tento kód treba najskôr skompilovať. Použijeme teda položku menu **Build – Build Solution**. Všimnime si, že v spodnom okne sa nám zobrazí protokol o kompilácii.

```
------ Build started: Project: MMITpokus, Configuration: Debug .NET ------
Preparing resources...
Updating references...
Performing main compilation...

Build complete -- 0 errors, 0 warnings
Building satellite assemblies...
---------------------- Done ----------------------

    Build: 1 succeeded, 0 failed, 0 skipped
```

Ak znovu navštívime URL adresu nášho projektu, po kompilácii je už všetko tak, ako má byť. Náš projekt, zatiaľ prázdny, ktorý pochopiteľne nemá nič robiť ani zobrazovať (a teda nemá robiť ani chyby) sa už podľa toho aj správa

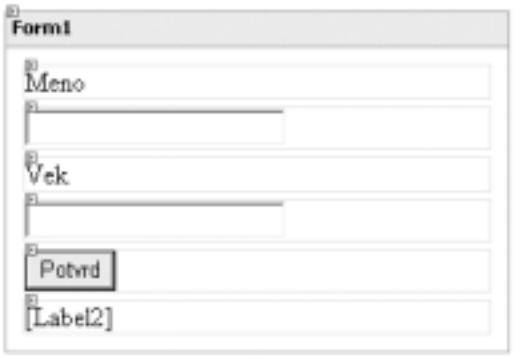
a výsledkom je prázdne okno zobrazené vo webovom prehliadači. Ak budeme pátrať po tom, kde je umiestnený skompilovaný kód, je v našom projektovom adresári v podadresári bin a v našom konkrétnom prípade je v súbore MMITpokus.DLL.

Teraz je už ale najvyšší čas stanoviť nejaké zadanie jednoduchej aplikácie. Bude to opäť klasika:

Cvičná aplikácia

Zadanie úlohy číslo 5.1: Vytvorte aplikáciu, ktorá umožní používateľovi zadať meno a vek a následne tieto údaje vypíše na obrazovke.

Riešenie úlohy číslo 5.1: Prvá cvičná aplikácia bude aj v tomto prípade jednoduchý formulár pre zadanie mena a veku. Po zatlačení tlačidla potvrď budú tieto údaje vypísané na obrazovku Z arzenálu komponent ponúkaného Toolboxom tri najjednoduchšie, ale asi aj najpoužívanejšie komponenty Label, Textbox a Button. Ale POZOR. Všimnite si, že uvedené komponenty sa nachádzajú dokonca až v troch záložkách. Buď ako komponenty klasických HTML stránok, alebo ako komponenty typu Web Forms, alebo Mobile Web Forms. Použijeme samozrejme komponenty typu Mobile Web Forms.



Obrázok 5.14 – Jednoduchý formulár pre zadávanie údajov

Ak chceme otestovať našu stránku, zostavíme projekt a zatlačíme tlačidlo **Potvrď**, nestane sa zatiaľ vôbec nič, ale takto to má byť, žiadny kód pre obsluhu tohoto tlačidla sme zatiaľ nenapísali.

Obrázok 5.15 – Aplikácia v okne emulátora prehliadača

Kódy pre obsluhu udalostí

V tomto okamihu nastal ten správny čas, aby sme sa začali zaujímať o obslužný kód. Visual Studio .NET pri návrhu projektu dôsledne dodržiava zásahu oddelenia návrhu formulárov od kódu. Návrh formulárov je v nám dobre známom súbore MobileWebForm1.aspx Naprotitomu všetok kód je sústredený v súbore **MobileWebForm1.aspx.cs**

```
..
namespace MMITpokus
{
    /// <summary>
    /// Summary description for MobileWebForm1.
    /// </summary>
    public class MobileWebForm1 : System.Web.UI.MobileControls.MobilePage
    {
      protected System.Web.UI.MobileControls.Form Form1;

      private void Page_Load(object sender, System.EventArgs e)
      {
        // Put user code to initialize the page here
      }

      #region Web Form Designer generated code
      override protected void OnInit(EventArgs e)
      {
        //
        // CODEGEN: This call is required by the ASP.NET Web Form Designer.
        //
        InitializeComponent();
        base.OnInit(e);
      }

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
        this.Load += new System.EventHandler(this.Page_Load);
      }
      #endregion
    }
}
```

Ak si kód pozrieme pozornejšie, vidíme, že vložené komentáre sa nás snažia navigovať. Do tela metódy **Page_Load** môžeme napríklad dopísať kód, ktorý je potrebné vykonať pri inicializácii stránky a podobne.

```
...
public class MobileWebForm1 : System.Web.UI.MobileControls.MobilePage
    {
      protected System.Web.UI.MobileControls.Label Label1;
      protected System.Web.UI.MobileControls.TextBox TextBox1;
      protected System.Web.UI.MobileControls.Command Command1;
      protected System.Web.UI.MobileControls.Label Label2;
      protected System.Web.UI.MobileControls.Form Form1;
...
```

Vidíme, že návrhová časť Visual Studia sa činila. Jednak v etape návrhu aplikácie, aj počas nášho návrhu aplikačného prostredia, pri ktorom sme rozmiestňovali komponenty webového formulára. Teraz už nie je záhadou, ani to, prečo sa po zatlačení tlačidla Potvrď nestalo vôbec nič. Obslužná procedúra Button1_Click neobsahuje totiž žiadny kód.

```
private void Command1_Click(object sender, System.EventArgs e)
{

}
```

Ak chceme len vypísať obsah editačného okienka v komponente Label3, dopíšeme do procedúry kód

```
private void Command1_Click(object sender, System.EventArgs e)
{
    Label3.Text = "Meno " + TextBox1.Text + ", Vek " + TextBox2.Text;
}
```

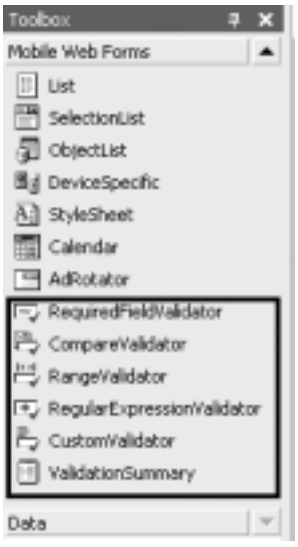
Ak v tomto okamihu otestujeme funkčnosť našej cvičnej miniaplikácie, dopadne to celkom úspešne.

Obrázok 5.16 – Aplikácia v okne emulátora prehliadača

Našu webovú aplikáciu môžeme samozrejme vyskúšať aj pomocou emulátora Pocket PC 2002 v okne internetového prehliadača

Obrázok 5.17 – Aplikácia v okne emulátora prehliadača Pocket PC2002

Ako ukážka dobré, no v reálnej praxi obvykle údaje zadané používateľom rôzne kontrolujeme, preverujeme a podobne. Ak napríklad mailová adresa neobsahuje zavináč, je zbytočné s ňou ďalej pracovať, u niektorých políčok požadujeme aby boli vyplnené nutne... Proste snažíme sa všetko čo sa dá skontrolovať. U technológie ASP.NET pre tento účel slúžia komponenty zvané validátory. Vo Visual Studiu s nimi pracujeme podobne ako s ostatnými komponentami Toolboxu.



Obrázok 5.18 – Ponuka validátorov

RequiredFieldValidator

Ešte predtým, než budeme kontrolovať, či používateľ zadal správne údaje, niekedy potrebujeme overiť ešte oveľa triviálnejšiu vec, totiž či používateľ vôbec nejaké údaje zadal. Hodí sa to napríklad pri zadávaní mena, kde iné kontrolné kritériá pravdepodobne ani neprichádzajú do úvahy.

CompareValidator

Tento validátor kontroluje zadanú hodnotu s referenčnou, prípadne s hodnotou iného poľa. Notoricky známym a v praxi veľmi často používaným príkladom je overovanie hesla pri jeho zadaní, či sme sa napríklad nedopustili preklepu, ktorý by nám mohol v budúcnosti veľmi znepríjemniť život.

RangeValidator

U číselných hodnôt veľmi často potrebujeme skontrolovať, či je zadaná hodnota v určitom číselnom intervale, ktorý je obvykle daný aplikačnou logikou.

RegularExpression Validator

Ako vyplýva z názvu, tento validátor slúži pre overenie, či zadaná hodnota vyhovuje regulárnemu výrazu. Typickým príkladom regulárneho výrazu je napríklad mailová adresa.

Všeobecný syntaktický predpis pre použitie **RegularExpression** validátora je nasledovný:

```
<asp:RegularExpressionValidator
      id="ID validátora"
      ControlToValidate=" ID prvku, ktorého obsah chceme kontrolovať"
      ValidationExpression="výraz"
      ErrorMessage="Oznam, ktorý sa vypíše v prípade neplatnej validácie"
      Text="text prvku"
      ForeColor="hodnota"
      BackColor="hodnota" …
      runat="server">
</asp: RegularExpressionValidator>
```

Predpis pre overenie regulárneho výrazu sa vytvára pomocou kombinácie špeciálnych znakov. Ich kompletný zoznam je v dokumentácii. Vo forme tabuľky ukážeme niekoľko najpoužívanejších.

Znak	Popis
*	Žiadny znak, prípadne viac znakov.
+	Jeden alebo viac znakov.
\	Prefix pred špeciálnym znakom, napríklad '\n' znamená znak CR. Používa sa aj pre aplikáciu špeciálnych znakov ako bežných
.	Jeden ľubovoľný znak okrem CR.
\d	Číslica.
\D	Iný znak ako číslica
\t	Tabulátor.

Kompletná tabuľka obsahuje niekoľko desiatok riadkov, no popis bez praktickej ukážky by nebol príliš účelný. Ukážeme to radšej na niekoľkých príkladoch. Predpis pre kontrolu mailovej adresy bude:
'.*@.*\..*', po preklade do ľudskej reči to znamená:

. jeden ľubovoľný znak
* ľubovoľný počet znakov (aj žiadne).
@ konkrétny znak, v našom prípade populárny zavináč
. jeden ľubovoľný znak
* ľubovoľný počet znakov (aj žiadne).
\. konkrétny znak, v tomto prípade bodka. Pretože bodka je aj špeciálny znak, je pred ňou lomítko
. jeden ľubovoľný znak
* ľubovoľný počet znakov (aj žiadne).

Zatiaľ to súvislú reč nepripomína (skôr nesúvislé myšlienky po nejakom „dobrom“ večierku). Ak by sme z toho chceli vykúzliť súvislú a logickú vetu, táto by znela. Reťazec začína jedným, alebo viacerými znakmi, za nimi musí byť znak zavináč, pokračujeme ďalším minimálne jednoznakovým reťazcom za ktorým musí byť bodka. Mailová adresa končí ukončovacím, minimálne jednoznakovým reťazcom. Jednoduché že? Ale stavím sa, že keby ste presne nevedeli o čo sa v prípade mailovej adresy jedná, ani z tohoto výkladu by ste to na prvý krát nepochopili. Po preštudovaní pár príkladov to už bude podstatne jasnejšie. Najjednoduchší a pritom veľmi užitočný a často používaný prípad pre validátor je kontrola mailovej adresy. Už vieme, že predpis pre kontrolu spomínaných pravidiel bude:
`ValidationExpression='.*@.*\..*'`.

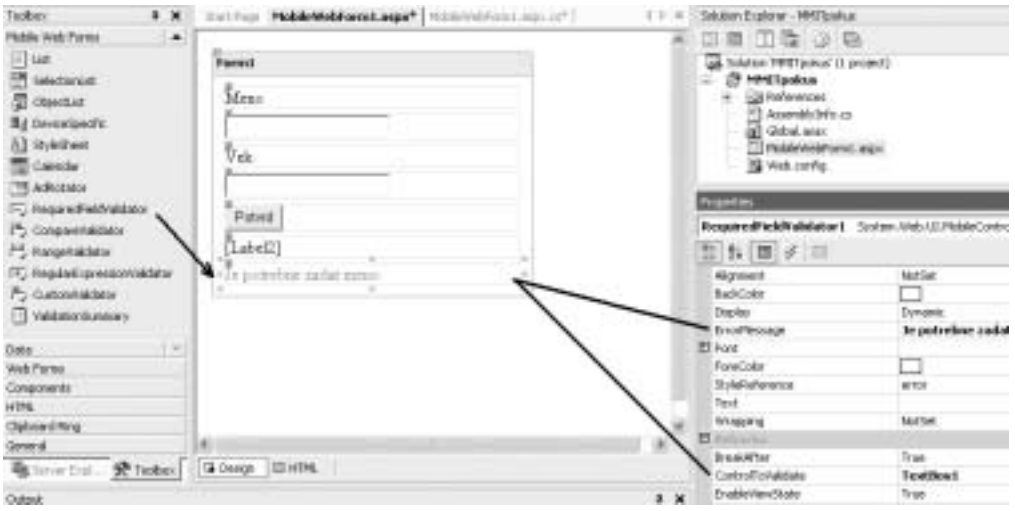
Custom Validator

Doteraz preberané validátory boli svojimi tvorcami navrhnuté tak, aby pomáhali validovať najčastejšie sa vyskytujúce situácie. Veľakrát sme postavení pred situáciu, validovať nejakú súvislosť, ktorá vyplýva z aplikačnej logiky. Potrebujeme napríklad skontrolovať rodné číslo, číslo kreditnej karty a podobne. Napríklad Rodné číslo v Českej aj Slovenskej republike sa zadáva v tvare napríklad 650327/6153. Kontrolný mechanizmus u rodných čísel osôb po roku asi 1953 spočíva v kontrole deliteľnosti celého rodného čísla jedenástimi. Podobné závislosti platia aj pre základnú kontrolu čísla kreditnej karty a podobne. Nejedná sa o nijakú podrobnú kontrolu a už vôbec nie validáciu, no už aj táto jednoduchá kontrola v prípade rodného čísla s pomerne vysokou pravdepodobnosťou odhalí náhodne zadané rodné číslo a samozrejme pomôže aj odhaliť preklep pri jeho zadávaní.

ValidationSummary Control

Ak navrhujeme formulár na ASP.NET stránke, ktorý má klient vyplniť, tento formulár obsahuje spravila viac editačných okien (chceme sa predsa od klienta dozvedieť čo najviac), a každé z nich môže byť ošetrené validátorom iného typu. Pomocou prvku ValidationSummary dokážeme vypísať oznamy všetkých validátorov na jednom mieste.

V našej aplikácii použijeme **RequiredFieldValidator**. pre overenie, či používateľ zadal meno. Požadovaný validátor jednoducho metódou drag and drop presunieme na vhodné miesto návrhového formulára. V ľavom dolnom okne pracovnej obrazovky nastavíme jeho vlastnosti, napríklad farbu, text chybového hlásenia a podobne. Nutne musíme nastaviť parameter **ControlToValidate**. Aj tento úkon je však zjednodušený na maximum. Stačí vybrať zo zoznamu použitých validovateľných komponentov ten správny, v našom prípade TextBox1.



Obrázok 5.19 – použitie validátora

Funkčnosť validátora vyskúšame veľmi jednoducho. Stačí ponechať políčko pre zadanie mena prázdne a zatlačiť tlačidlo Potvrď.

Obrázok 5.20 – overenie validátora

Možnosti webových aplikácií sú hlavne v spolupráci s databázovým serverom takmer neobmedzené (jedno obmedzenie tu samozrejme je – musíme byť on-line), preto si pre druhú cvičnú aplikáciu stanovíme náročnejší cieľ

Vytvorenie databázovej aplikácie pomocou MMIT

Zadanie úlohy číslo 5.2: Vytvorte webovú aplikáciu typu adresár, ktorá umožní používateľovi vyhľadávať údaje o osobách uložených v databáze. Využite údaje z cvičnej databázy Northwind. Pre využitie vlastností mobilného zariadenia pridajte do aplikácie odkaz na hot-line Microsoftu.

Riešenie úlohy číslo 5.2: Najskôr urobíme sumár, čo potrebujeme mať na počítači na ktorom túto aplikáciu budeme vyvíjať nainštalované:

- Internet Information Services (IIS) a .NET Framework
- Visual Studio .NET
- Microsoft Mobile Internet Toolkit
- Microsoft Mobile Explorer 3.0 prípadne aj Microsoft Pocket PC emulator
- Databázový server SQL Server 2000 a cvičnú databázu NorthWind

Vo Visual Studiu vytvoríme novú aplikáciu typu **Mobile Web Application** s názvom NWadresar. Adresa pre ladenie projektu potom bude *http://localhost/NWadresar* Riešenie rozdelíme do niekoľkých krokov.

Krok1. – vytvorenie formulára aplikácie.

- na plochu aplikačného formulára umiestnime komponentu **Label** s textom **Telefonny zoznam** a nastavíme parameter StyleReference = title
- na plochu aplikačného formulára umiestnime komponentu **PhoneCall** s textom **Hotline Microsoftu** a nastavíme parametre `PhoneNumber=1-800-555-1212` a `AlternateUrl=http://www.Microsoft.com` .
- pod aplikačný formulár Form1 umiestnime ďalšiu komponentu **Form**. Táto bude mať názov Form2.
- na plochu aplikačného formulára Form1 umiestnime komponentu **Link** s textom **Najdi pracovníka** Parameter **NavigateURL** nastavíme na hodnotu **#Form2**

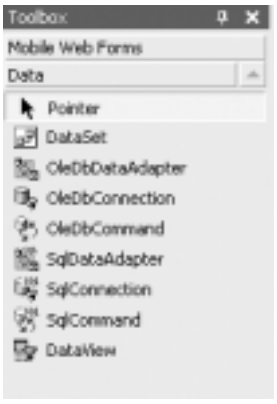
Obrázok 5.21 – Návrh aplikačného formulára

V závere prvého kroku môžeme aplikáciu zostaviť pomocou menu **Build | Build Solution** a vyskúšať.

Obrázok 5.22 – Test aplikácie v rôznych emulátoroch

Krok2. – Pripojenie sa k databáze pod SQL Server 2000 prostredníctvom prvku dataset.

V Toolboxe zvolíme záložku **Data** a presunieme na pracovnú obrazovku prvok **SQLDataAdapter**.



Obrázok 5.23 – Komponenty Toolboxu zo záložky Data

Po umiestnení prvku **SQLDataAdapter** sa automaticky aktivuje sprievodca pripojení k databázovému serveru. Pomocou tlačidla **New Connection** vytvoríme pripojenie k databáze Northwind.

Obrázok 5.24 – parametre pre pripojenie k databáze Northwind

Pomocou tlačidla **Test Connection** môžeme toto pripojenie otestovať. V nasledujúcom dialógu nastavíme voľbu **Use SQL statements**. Do ďalšieho okna zadáme SQL dotaz pre výber údajov z databázy:

```
SELECT LastName, FirstName, Title, Address, City, Region, Country, Extension
FROM Employees
```

Klikneme na ikonu umiestnení prvku **SQLDataAdapter** pod formulárom a v menu Visual Studia vyberieme položky **Data | Generate Dataset** a do editačného poľa **New** zadáme: **employeeDataSet**. Zaškrtneme položku **Add this dataset to the designer**.

V okne Solution Explorer dvakrát klikneme na položku **MobileWebForm.aspx.** a v menu View klikneme na položku **Code**.

Do procedúry **Page_Load** pridáme kód (vyznačený hrubým fontom)

```
private void Page_Load(object sender, System.EventArgs e)
{
    // Put user code to initialize the page here
    sqlDataAdapter1.Fill(employeeDataSet, "Employees");
    Page.DataBind();
}
```

V tejto fáze môžeme zostaviť aplikáciu (pomocou menu **Build | Build Solution**), aby sme si overili, či sme sa nedopustili nejakej chyby, alebo preklepu, no zatiaľ sa ju nepokúšajme spustiť. Aplikácia by totiž vygenerovala chybové hlásenie, že sa nepodarilo pripojiť klienta ASPNET k serveru. Databázový server je totiž z bezpečnostného hľadiska navrhnutý tak, že neumožní nič, čo sme predtým sami nepovolili. Takže musíme v SQL Serveri 2000 vytvoriť nového klienta, v našom prípade ASPNET a prideliť mu privilégiá do databázy Northwind, konkrétne do tabuľky **Employees**.

Obrázok 5.25 – Vytvorenie nového používateľa pre SQL Server

Pomocou nástroja **SQL Server Enterprise manager** (aktivujeme ho z hlavného menu Windows v zložke Microsoft SQL Server) vytvoríme nového používateľa **ASPNET** s prístupovými privilégiami Windows NT. V našom prípade používateľovi ASPNET povolíme v databáze Northwind, konkrétne v tabuľke Employees vykonávať príkazy **SELECT, INSERT, UPDATE** a **DELETE**.

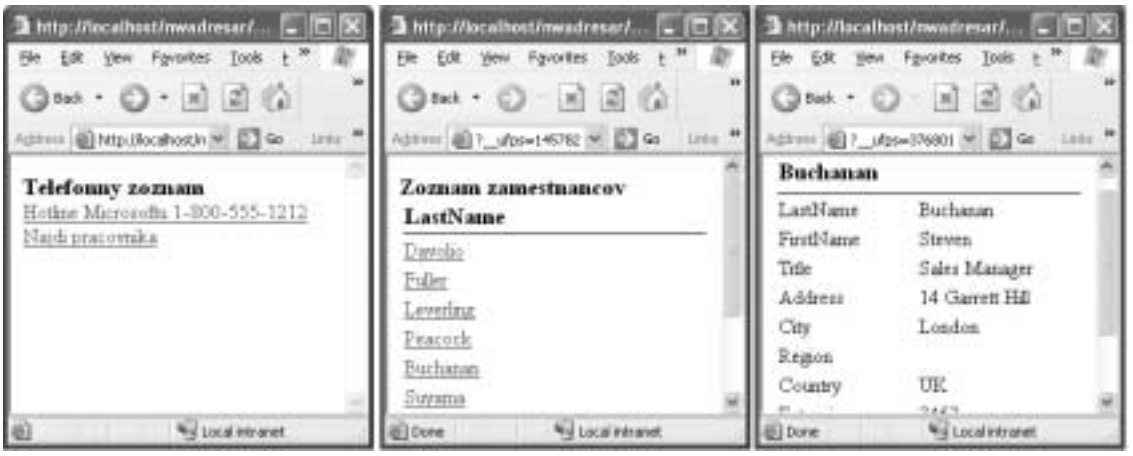
Na plochu formulára **Form2** umiestnime komponentu **Label** s textom **Zoznam zamestnancov** a nastavíme parameter StyleReference = title

Na plochu formulára **Form2** umiestnime komponentu **ObjectList** a nastavíme parametre
DataSource = employeeDataSet1
DataMember = Employees
LabelField = LastName

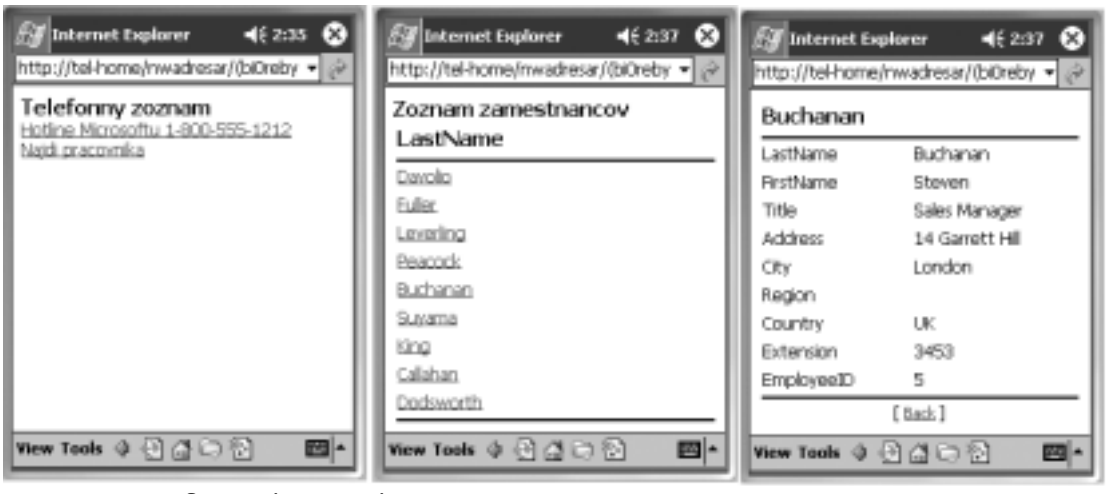
Pre formulár Form2 nastavíme parameter **Paginate** na hodnotu **TRUE**. Týmto krokom je návrh formulára predbežne ukončený

Obrázok 5.26 – Návrh aplikačného formulára

Nakoniec môžeme pomocou rôznych prehliadačov a emulátorov vychutnávať plody svojej práce. Začneme Internet Explorerom pod operačným systémom Windows XP, (okno sme úmyselne zmenšili z dôvodu úspory miesta v zborníku)



Pokračujeme emulátorom platformy POCKET PC 2002



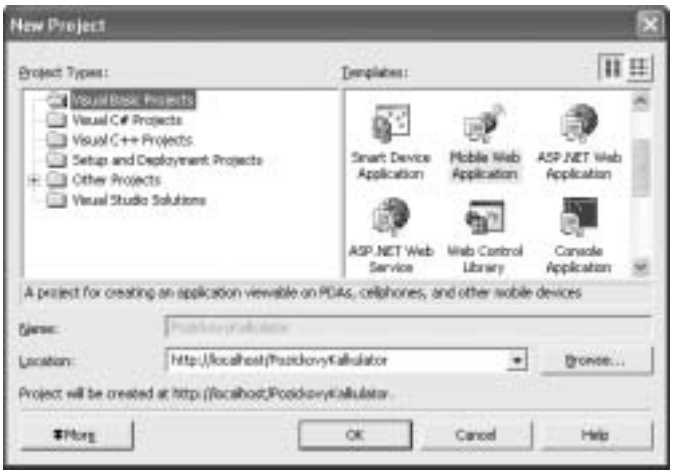
A nakoniec vyskúšame beh aplikácie pomocou simulátora **Openwave** a to na dvoch typoch zariadení

Vytvorenie aplikácie ktorá bude využívať webovú službu pomocou MMIT (Visual Basic)

Zadanie úlohy číslo 5.3: Vytvorte webovú aplikáciu, ktorá spočíta výšku mesačnej splátky pri zadaní výšky pôžičky, úrokovej miery a počtu mesiacov na ktoré je pôžička poskytnutá. Aplikácia bude využívať webovú službu **wsSplatky**.

Riešenie úlohy číslo 5.3: Pred začiatkom vývoja aplikácie predpokladáme, že máme na serveri nainštalovanú webovú službu wsSplatky.

Vo Visual Studiu.NET vytvoríme novú aplikáciu typu **Mobile Web Application** (zložka **Visual Basic Projects**) s názvom **PozickovyKalkulator**. Adresa pre ladenie projektu potom bude *http://localhost/PozickovyKalkulato*r



Obrázok 5.30 – Nový projekt – aplikácia typu Mobile Web Application

Registrácia XML webovej služby

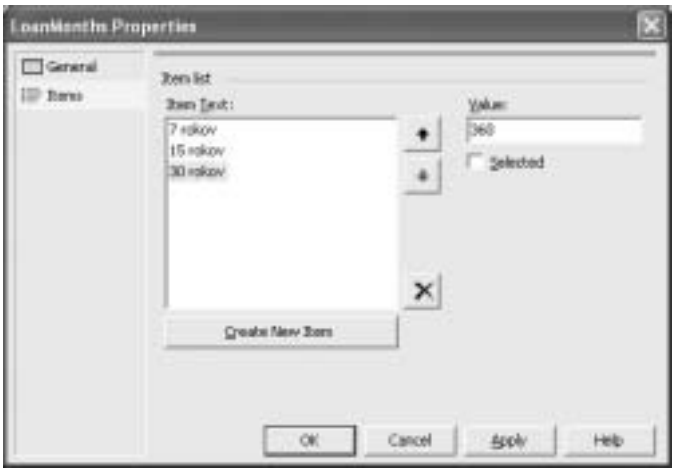
V nasledujúcom kroku zaregistrujeme XML webovú službu. Klikneme pravým tlačidlom myši na položku **References** v okne **Solution Explorer** a v zobrazenom menu klikneme na položku **Add Web Reference**. V dialógu, ktorý sa následne otvorí zadáme referenciu na webovú službu, v našom prípade adresu *http://localhost/wsSplatky/wsSplatky.asmx* Tlačidlom Add Reference pridáme do projektu referenciu na webovú službu

Obrázok 5.31 – Registrácia webovej služby

Vytvorenie formulára pre komunikáciu s webovou službou

V tomto kroku vytvoríme aplikačný formulár, ktorý bude slúžiť ako používateľské rozhranie z mobilného zariadenia na webovú službu. Do formulára **Form1** pridáme komponenty:

Label – parameter **text** nastavíme na **Výška pôžičky**
TextBox – parameter **ID** nastavíme na **LoanAmount**
Label – parameter **text** nastavíme na **Úrok**
TextBox – parameter **ID** nastavíme na **LoanRate**
Label – parameter **text** nastavíme na **Pocet rokov**
SelectionList – parameter **ID** nastavíme na **LoanMonths**
parameter **SelectType** nastavíme na **Radio**
parameter **Items** klikneme na tlačidlo **[...]**

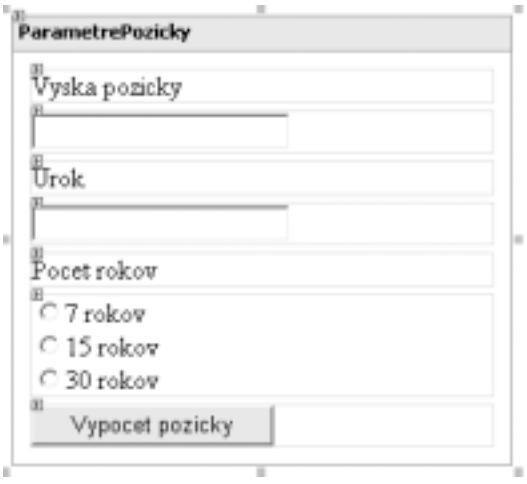


Obrázok 5.32 – Parametre pre komponentu Selection List

V dialógu nastavíme hodnoty položiek, počet mesiacov pre stanovené obdobia 7, 15 a 30 rokov
- 7 rokov – 84
- 15 rokov – 180
- 30 rokov – 360

Pridáme komponentu **Command**
– parameter **text** nastavíme na **Výpočet pôžičky**
– parameter **ID** nastavíme na **CalculateLoan**
– parameter **SoftKeyLabel** nastavíme na **Vypocet**

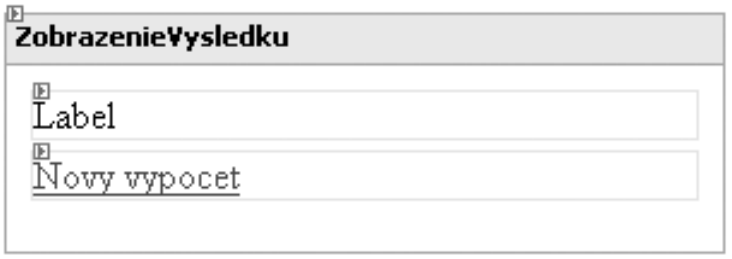
Premenujeme **ID** formulára **Form1** na **ParametrePozicky**



Obrázok 5.33 – Návrh formulára ParametrePozicky

Na pracovnú plochu pod formulár **ParametrePozicky** pridáme novú komponentu typu Form (zatiaľ bude má ID Form1). Premenujeme ho na **ZobrazenieVysledku**.

Do nového formulára pridáme komponenty
Label – parameter **ID** nastavíme na **LoanResult**
Link – parameter **NavigateUrl** nastavíme na #**ParametrePozicky**
parameter **text** nastavíme na **Nový výpočet**



Obrázok 5.34 – Návrh formulára ZobrazenieVysledku

Obsluha udalosti – volanie webovej služby

Udalosť, ktorú v tomto prípade chceme ošetriť je jednoznačná – kliknutie na tlačidlo Výpočet pôžičky. Po dvojkliku na jeho symbol sa zobrazí telo procedúry **`CalculateLoan`** (zatiaľ prázdne)

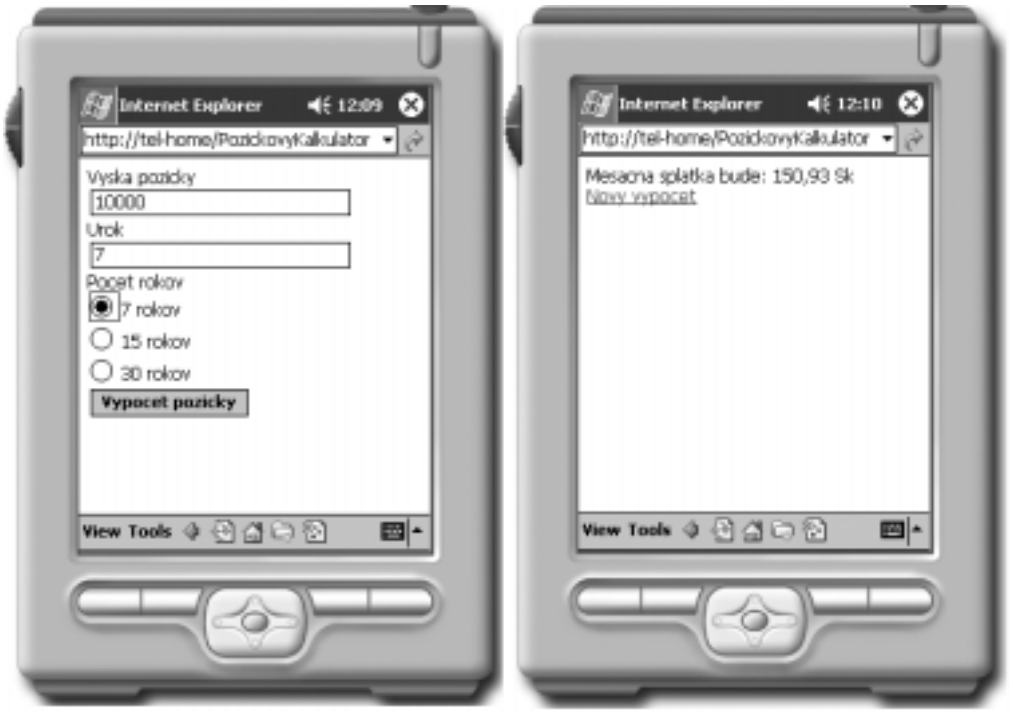
```
Private Sub CalculateLoan_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CalculateLoan.Click

End Sub
```

Do tela tejto procedúry pridáme kód:

```
        'inicializacia XML Web Service proxy
        Dim wsSplatky As New localhost.Service1()
        ' Deklaracia retazca pre zobrazenie vysledku
        Dim monthlyPayment As String
        ' Volanie webovej sluzby, metoda CalculateMonthlyPayment
        monthlyPayment = wsSplatky.CalculateMonthlyPayment(LoanAmount.Text,
LoanMonths.Selection.Value, LoanRate.Text)
        LoanResult.Text = "Mesacna splatka bude: " + monthlyPayment
        ActiveForm = ZobrazenieVysledku
```

Po úspešnom preklade a zostavení webovej aplikácie môžeme túto otestovať pomocou prehliadačov a emulátorov. Adresa je v našom prípade *http://localhost/PozickovyKalkulator/MobileWebForm1.aspx*

Obrázok 5.35 – Fungovanie aplikácie na rôznych typoch zariadení



Obrázok 5.36 – Fungovanie aplikácie na emulátore Pocket PC 2002

V projekte môžeme ďalej pokračovať, napríklad pridať validátory a podobne.

Kapitola 6

Vývoj aplikácií pre Smartphone

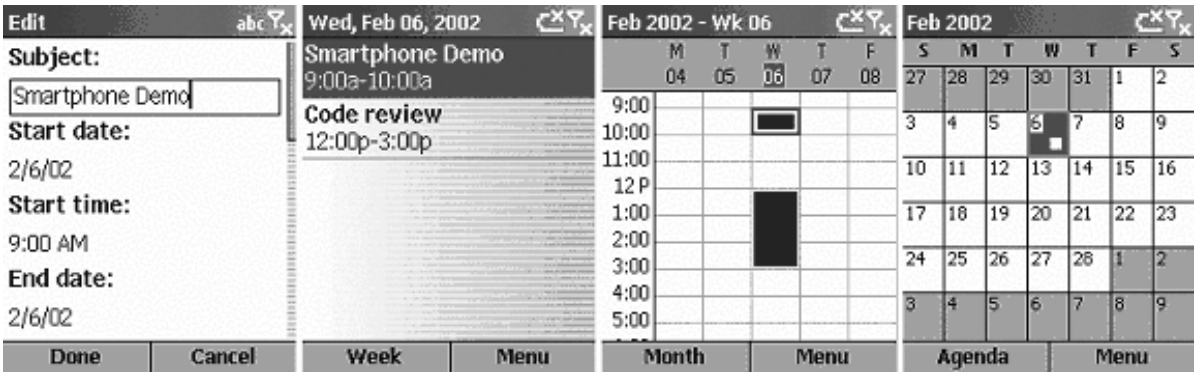
Predmetom tejto kapitoly bude vývoj aplikácií pre platformu Smartphone pomocou **Microsoft Windows SDK Beta for Smartphone 2002**. V prvej verzii je k dispozícii pre natívny vývoj, t.j zatiaľ nie je možné na SmartPhone spúšťať managed kód, no v budúcnosti sa bude vývoj uberať práve touto cestou. Platforma **Smartphone 2002** nakoľko je implementovaná v mobilných telefónoch má určité špecifiká, hlavným rozdielom oproti Pocket PC 2002 je absencia dotykového displeja. Mobilné telefóny sa totiž ovládajú hlavne klávesnicou a aj tá býva počas nosenia vo vrecku zablokovaná, takže dotykový displej bez odklopného krytu by bol nezmysel. Displej bez možnosti ovládania dotykom môže mať jednak odolné čelné sklo, ale zníži sa tým aj cena zariadenia a jeho spotreba. Typický prístroj triedy Smartphone má približne takýto dizajn

Obrázok – Typický dizajn Smartphone 2002

Pri pohľade na prístroj nie je zložité vymenovať všetky jeho ovládacie možnosti.

Klávesy 0-9,
#,
***,**
Action,
Up,
Down,
Right,
Left,
End,
Send,
Soft Key 1,
Soft Key 2,
Volume Up,
Volume Down,
Home,
Back,
Power

Korešponduje to s udalosťami, ktoré nastanú používateľovým pričinením pri komunikácii používateľa s prístrojom. Rozlíšenie displeja je 176 x 220 pixelov. Displej môže byť farebný alebo čiernobiely. O zobrazovacích možnostiach pri tomto zdanlivo nízkom rozlíšení vypovedá nasledujúci obrázok.



Obrázok – Zobrazovacie možnosti Smartphone 2002

Pri návrhu aplikácie môžeme využiť komponenty:

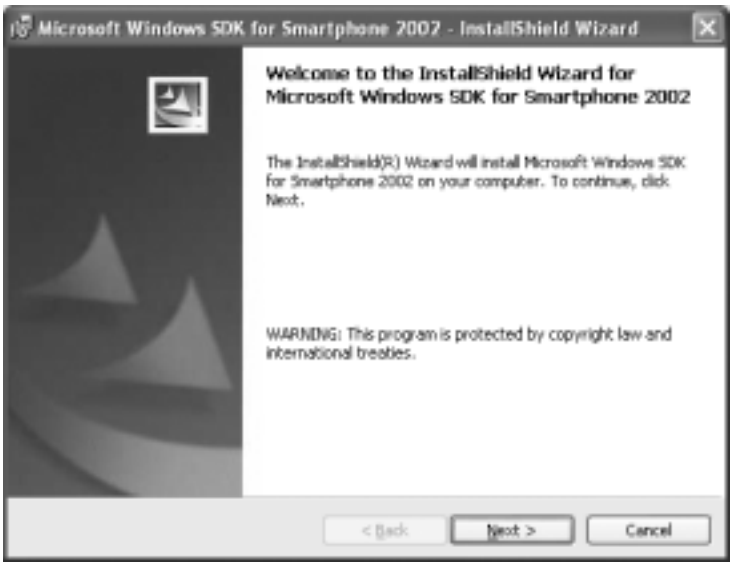
Edit Control
Radio Buttons (len Pocket IE)
Check boxes
Soft Keys
Spinners
Tree View
Date Picker
Date Picker
Horizontal Line Separator
Progress Meter
Alert and Message Box
Scroll Bars
Tabs

Vymenujeme aj technológie, ktorých možnosti môžeme pri vývoji aplikácií využiť

ActiveSync
Windows CE Messaging
Connection Manager
Control API
Game API (GAPI)
Home Screen API
HTML Control
MIDI
Object Exchange (OBEX)
Pocket Outlook® Object Model (POOM) API
Projects Control
Remote API (RAPI)
Speech Recognizer
Telephony
User Interface
Voice Recorder Control

Inštalácia Microsoft® Windows® SDK for Smartphone 2002

Tento produkt je možné stiahnuť z webu Microsoftu, zo sekcie Download, alebo z adresy *http://www.microsoft.com/mobile/partners/smartphone* . Veľkosť inštalačného súboru je okolo 48MB. Predtým musíme mať nainštalované vývojové prostredie eMbedded Visual Tools 3.0. Toto vývojové prostredie (podrobne popísané v druhej kapitole) môžeme získať na adrese http://www.microsoft.com/mobile/developer/downloads/emvt30/



Obrázok – Inštalácia SDK for Smartphone 2002

V nasledujúcom dialógu sa môžeme rozhodnúť pre implicitnú inštaláciu, kedy sa nainštalujú všetky súčasti produktu, prípadne pre voliteľnú inštaláciu.

Obrázok – Voliteľná inštalácia MMIT

Aplikácia sa implicitne nainštaluje do adresára C:\Windows CE\wce300\Smartphone 2002. Súčasťou inštalácie je aj kvalitná dokumentácia, ktorá obsahuje **The Smartphone Design Guide** a **The Smartphone Porting Guide** pre portovanie aplikácií z platformy PocketPC pre platformu Smartphone.

Spustenie emulátora Pocket PC 2002 v režime Smartphone 2002 môžeme realizovať v adresári

```
C:\Program Files\Windows CE Tools\Common\Platman\bin\
```

trochu zložitejším príkazom

```
Emulator.exe /CEImage "C:\Windows CE Tools\wce300\Smartphone 2002\emulation\Western
European-No Radio\WWENoRIL.bin" /Video 176x220x16 /Ethernet TRUE /MemorySize 32 /VMID
5F2AD04332BE4874B5B1B16758EC1B1D
```

Obrázok – Emulátor Pocket PC2002 v móde Smartphone 2002

Ak začneme po pracovnej ploche emulátora klikať kurzorom myši v domnení, že simulujeme dotykové pero, nebude sa nám dariť. Ako sme uviedli v úvode kapitoly, platforma Smartphone 2002 dotykové pero nepoužíva, všetko ovládanie je vyriešené klávesnicou mobilného telefónu. Tomu sa musíme prispôsobiť pri ovládaní emulátora. Pomôže nám táto tabuľka.

Phone Key	CEPC Key
Home	Windows key
Back	Escape
Softkey 1	F1
Softkey 2	F2
Send	F3
End	F4
*	F8
#	F9

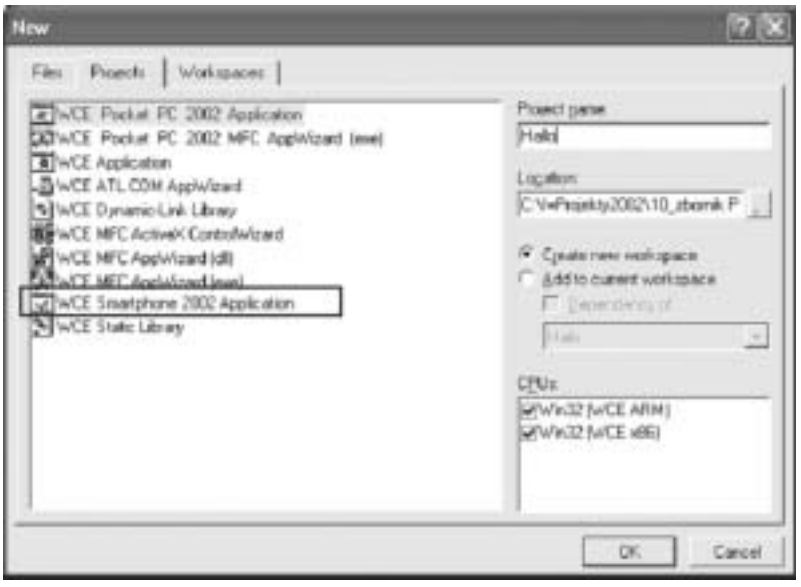
V nasledujúcej tabuľke je prehľad udalostí, ktorých programová obsluha prichádza do úvahy u aplikácií pre platformu Smartphone. Jedná sa o udalosti generované klávesnicou. Rozmiestnenie numerických kláves aj spôsob zadávania alfanumerických znakov z numerickej klávesnice poznáme dôverne z bežných mobilných telefónov. V stĺpci výsledok je pre jednotlivé číslice presný význam „Generuje DTMF kód (kód telefónnej tónovej voľby) pre číslicu X“

Názov	Kód	Vysledok	Zatlačené podržané
0	VK_0	Číslica 0. „Next word“ pre T9	Speed dial
1	VK_1	Číslica 1	Speed dial, voice mail
2	VK_2	Číslica 2 a znaky „A“, „B“, „C“, „a“, „b“, „c“.	Speed dial
3	VK_3	Číslica 3 a znaky „D“, „E“, „F“, „d“, „e“, „f“.	Speed dial
4	VK_4	Číslica 4 a znaky „G“, „H“, „I“, „g“, „h“, „i“.	Speed dial
5	VK_5	Číslica 5 a znaky „J“, „K“, „L“, „j“, „k“, „l“.	Speed dial
6	VK_6	Číslica 6 a znaky „M“, „N“, „O“, „m“, „n“, „o“.	Speed dial
7	VK_7	Číslica 7 a znaky „P“, „Q“, „R“, „S“, „p“, „q“, „r“, „s“.	Speed dial
8	VK_8	Číslica 8 a znaky „T“, „U“, „V“, „t“, „u“, „v“.	Speed dial
9	VK_9	Číslica 9 a znaky „W“, „X“, „Y“, „Z“, „w“, „x“, „y“, „z“.	Speed dial
#	VK_#	Znak „#“ a znak medzery „space“.	Zoznam symbolov
*	VK_*	Znak „*“, shift a caps lock.	Zmena vstupneho módu
Action	VK_ACTION	Analógia kliknutia tlačidlom myši u PC	
Down	VK_DOWN	Analógia klávesy „šípka nadol“	Rýchlejšie skrolovanie
End	VK_END	Reakcia na tlačidlo END. Niekedy býva označované ako „CANCEL“ alebo „NO“	Key lock
Send	VK_SEND	Reakcia na tlačidlo SEND. Niekedy býva označované ako „OK“	
Soft Key 1	VK_SOFT1	Tlačidlo s dynamicky priraďovaným významom 1	
Soft Key 2	VK_SOFT2	Tlačidlo s dynamicky priraďovaným významom 2	
Up	VK_UP	Analógia klávesy „šípka nahor“	Rýchlejšie skrolovanie
Volume Up	VK_VOLUP	Zvýšenie hlasitosti.	
Volume Down	VK_VOLDN	Zníženie hlasitosti.	
Left	VK_LEFT	Analógia klávesy „šípka vľavo“	Rýchlejšie skrolovanie
Right	VK_RIGHT	Analógia klávesy „šípka vpravo“	Rýchlejšie skrolovanie
Home	VK_HOME	Zobrazí Homepage zariadenia.	
Back	VK_BACK	Analógia klávesy BACKSPACE u PC	V editačnom móde všetko vymaže.
Power		Ak je zariadenie vypnuté, zapne sa. Pri zapnutom zariadení sa zobrazí menu pre rýchly prístup.	Zapne, alebo vypne zaiadenie.

Podrobnosti ohľadne vývoja aplikácií v prostredí **Microsoft® eMbedded Visual Tools 3.0** (ďalej len eMVT) boli prebrané v druhej kapitole. Môžeme teda pristúpiť priamo k vývoju prvej aplikácie.

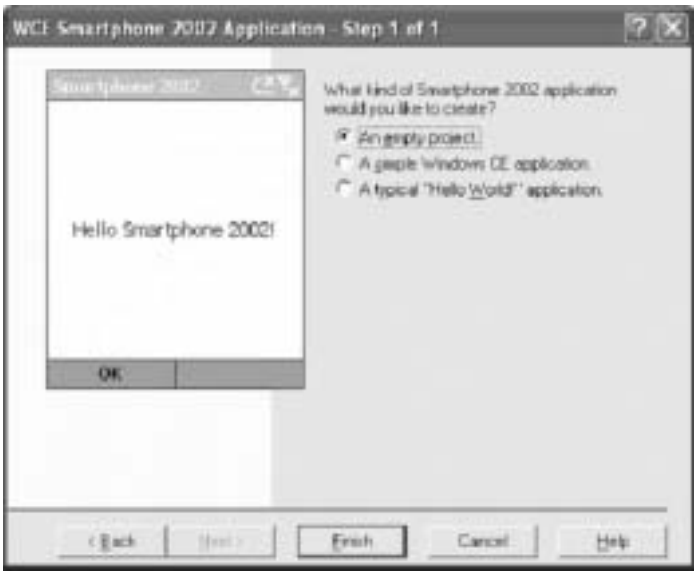
Zadanie úlohy 6.1: Vytvorte aplikáciu, ktorá bude demonštrovať výpis textu a kreslenie na displeji.

Riešenie úlohy 6.1: Pomocou menu **File | New** vytvoríme nový projekt typu **WCE Smartphone PC 2002 Applications**. V úvodnom dialógu aplikáciu pomenujeme, v našom prípade Halo.



Obrázok – Nová aplikácia pre platformu Smartphone 2002

Na výber máme tri typy aplikácie:
– prázdny projekt
– jednoduchá Windows CE aplikácia
– typická „Hello World“ aplikácia



Obrázok – Typy aplikácie pre platformu Smartphone 2002

Pri prvom projekte si necháme napovedať a preto zvolíme aplikáciu typu „Hello World“. Sprievodca vytvorením aplikácie (Aplikačný wizard) vygeneroval tento kód (trochu priestorovo zhustený)

```
// Halo.cpp : Defines the entry point for the application.
#include <windows.h>
#include <aygshell.h>
#include "resource.h"

HINSTANCE g_hInst = NULL;   // Local copy of hInstance
HWND hwndMain = NULL;       // Handle to Main window returned from CreateWindow
```

```
TCHAR szAppName[] = TEXT("Smartphone 2002 Application");
TCHAR szTitle[]   = TEXT("Smartphone 2002");
TCHAR szMessage[] = TEXT("Hello Smartphone 2002!");

//  FUNCTION: WndProc(HWND, unsigned, WORD, LONG)
//  PURPOSE:  Processes messages for the main window.
//  WM_COMMAND       - process the application menu
//  WM_PAINT  - Paint the main window
//  WM_DESTROY       - post a quit message and return

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wp, LPARAM lp)
{
     LRESULT          lResult = TRUE;
    HDC        hdc;
    PAINTSTRUCT       ps;
    RECT       rect;

    switch(msg)
    {
      case WM_CREATE:
        // create the menu bar
        SHMENUBARINFO mbi;
        ZeroMemory(&mbi, sizeof(SHMENUBARINFO));
        mbi.cbSize = sizeof(SHMENUBARINFO);
        mbi.hwndParent = hwnd;
        mbi.nToolBarId = IDR_HELLO_MENUBAR;
        mbi.hInstRes = g_hInst;
        if (!SHCreateMenuBar(&mbi)) {PostQuitMessage(0);}
          break;

      case WM_COMMAND:
        switch (wp)
        {
           case IDOK: SendMessage(hwnd,WM_CLOSE,0,0);break;
           default:   return DefWindowProc(hwnd, msg, wp, lp);
        } break;

      case WM_PAINT:
      {
        hdc = BeginPaint (hwnd, &ps);
        GetClientRect (hwnd, &rect);
        DrawText (hdc, szMessage, -1, &rect, DT_SINGLELINE | DT_CENTER | DT_VCENTER);
        EndPaint (hwnd, &ps);
      } break;

      case WM_CLOSE: DestroyWindow(hwnd); break;
      case WM_DESTROY:PostQuitMessage(0);break;
      default:         lResult = DefWindowProc(hwnd, msg, wp, lp);break;
    }
    return (lResult);
}

//  FUNCTION: InitInstance(HANDLE, int) Saves instance handle and creates main window

BOOL InitInstance (HINSTANCE hInstance, int CmdShow )
{
```

```
  g_hInst = hInstance;
  hwndMain = CreateWindow(szAppName, szTitle, WS_VISIBLE,CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT, NULL, NULL, hInstance, NULL );

   if ( !hwndMain ) {return FALSE;}
   ShowWindow(hwndMain, CmdShow );
   UpdateWindow(hwndMain);
   return TRUE;
}

//  FUNCTION: InitApplication(HANDLE) Sets the properties for our window.

BOOL InitApplication ( HINSTANCE hInstance )
{
   WNDCLASS wc;
   BOOL f;
   wc.style = CS_HREDRAW | CS_VREDRAW ;
   wc.lpfnWndProc = (WNDPROC)WndProc;
   wc.cbClsExtra = 0;
   wc.cbWndExtra = 0;
   wc.hIcon = NULL;
   wc.hInstance = hInstance;
   wc.hCursor = NULL;
   wc.hbrBackground = (HBRUSH) GetStockObject( WHITE_BRUSH );
   wc.lpszMenuName = NULL;
   wc.lpszClassName = szAppName;
   f = (RegisterClass(&wc));
   return f;
}

//  FUNCTION: WinMain(HANDLE, HANDLE, LPWSTR, int) Entry point for the application

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR lpCmdLine,
                   int CmdShow)
{
   MSG msg;
   HWND hHelloWnd = NULL;
   hHelloWnd = FindWindow(szAppName, szTitle);
   if (hHelloWnd) {SetForegroundWindow (hHelloWnd); return 0;}
   if ( !hPrevInstance ){if ( !InitApplication ( hInstance ) ){return (FALSE); }}
   if ( !InitInstance( hInstance, CmdShow )) {return (FALSE);}

   while ( GetMessage( &msg, NULL, 0,0 ) == TRUE )
   {
     TranslateMessage (&msg);
     DispatchMessage(&msg);
   }
   return (msg.wParam);\

}
```

Obrázok – Spustenie aplikácie Halo

Zobrazenie textu sa teda vykoná ako reakcia na udalosť **WM_PAINT**
Fyzický výpis textu uloženého v reťazci `szMessage` zabezpečí procedúra **WM_PAINT**.

```
case WM_PAINT:
{
    hdc = BeginPaint (hwnd, &ps);
    GetClientRect (hwnd, &rect);
    DrawText (hdc, szMessage, -1, &rect, DT_SINGLELINE | DT_CENTER | DT_VCENTER);
    EndPaint (hwnd, &ps);
} break
```

Ak chceme vypísať a vykresliť na obrazovku viac, umiestnime príslušný kód práve ako reakciu na udalosť **WM_PAINT**.

```
case WM_PAINT:
{
     RECT rcString, rcClient;
     POINT ptTrig[4];
     INT i, cy;
     GetClientRect (hwnd, &rcClient);
     hdc = BeginPaint (hwnd, &ps);

     //trojuholniky
     ptTrig[0].x = ptTrig[3].x = ptTrig[1].x = rcClient.left;
     ptTrig[0].y = ptTrig[3].y = rcClient.top;
     ptTrig[2].y = ptTrig[1].y = rcClient.bottom;
     ptTrig[2].x = rcClient.right;
     SelectObject(hdc, (HBRUSH)GetStockObject(GRAY_BRUSH));
     Polygon(hdc, ptTrig, 4);
     SelectObject(hdc, (HBRUSH)GetStockObject(BLACK_BRUSH));
     ptTrig[0].x = ptTrig[3].x = ptTrig[2].x /2;
     ptTrig[0].y = ptTrig[3].y = ptTrig[2].y /3;
     Polygon(hdc, ptTrig, 4);
     SelectObject(hdc, (HBRUSH)GetStockObject(WHITE_BRUSH));
```

```
//vypis textu
      DrawText (hdc, TEXT ("Sample DrawText"), -1, &rcString, DT_CALCRECT | DT_CENTER |
                                DT_SINGLELINE);
      cy = rcString.bottom - rcString.top + 5;

      rcClient.bottom = rcClient.top + cy;
      SetBkMode (hdc, TRANSPARENT);
      for (i = 0; i < 4; i++)
         {
       SetTextColor (hdc, PALETTEINDEX2BPP(i));
       SetBkColor (hdc, PALETTEINDEX2BPP(3 -i));
       DrawText (hdc, TEXT ("Ahoj ja som Smartphone2002"), -1, &rcClient, DT_CENTER |
                        DT_SINGLELINE);
       rcClient.top += cy;
       rcClient.bottom += cy;
    }

      EndPaint (hwnd, &ps);
}
```

Znovu si prezrieme výsledok našej práce na displeji emulátora.

Obrázok - Spustenie aplikácie Halo

Vydal: Microsoft, s.r.o., Novodvorská 1010/14B, 140 00 Praha 4, tel.: +420 261 197 111, fax: +420 260 197 100
http://www.microsoft.com/cze, http://www.microsoft.com/slovakia