Microsoft .net

Přehled architektury .NET

Microsoft

Microsoft® Corporation

Přehled architektury .NET

Microsoft®

OBSAH

Kapitola 1:

Posun typových řešení k distribuovaným výpočtům po Internetu

Abstrakt

Tato kapitola zkoumá, jak posun k modelu distribuovaným výpočtům v prostředí Internetu podněcuje potřeby nových technologií, nových nástrojů a nových přístupů při vývoji aplikací. Diskutuje také podmínky trhu a nové technologie, které se staly základem pro platformu .NET.

ÚVOD

Článek popisuje, jak rychlý vývoj, všudypřítomnost a široké rozšíření Internetu umožnily vznik nového typu distribuovaných aplikací, které slibují nejen transformaci IT průmyslu a způsobu jakým řeší problémy tvorby softwarových systémů, ale také způsoby, jimiž společnosti provozují obchodní procesy v rámci digitální ekonomiky. Článek také pojednává o pozadí .NET platformy, tržních podmínkách a technologických zdokonaleních, které se staly jejím základem.

TECHNOLOGICKÉ TRENDY

Existence nového architektonického přístupu, podstatně odlišného od čehokoli co dnes máme k dispozici, je nutností při realizaci nové generace internetově orientovaných aplikací. Tyto aplikace budou bezproblémově navzájem spolupracovat a poskytovat uživatelům potřebné individuální informace a to kdykoli, kdekoli a z libovolného zařízení.

Tradiční model založený na prohlížeči, na který se mnoho vývojářů aplikací zaměřilo během předchozích pěti let, postupně stárne. Vzniká potřeba progresivních vývojových nástrojů a standardů společně s novými modely pro vytváření nové generace internetových aplikací.

Před tím, než se zaměříme na to, co nás čeká, podívejme se, jak se rozvinuly tradiční modely během různých ér počítačové technologie.

Centralizace

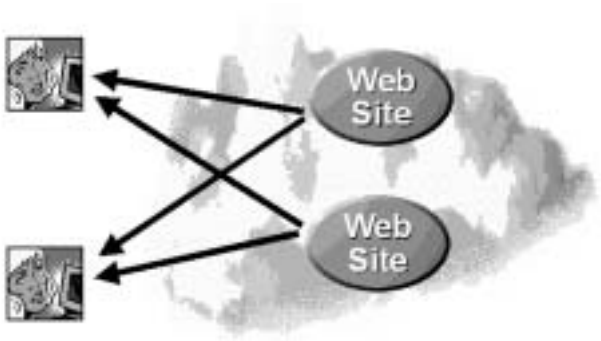
V 60. letech, před zrodem personálních počítačů, byl svět výpočetní techniky řízen mainframy. Vzdálené terminály poskytující uživatelské rozhraní byly ve své funkcionalitě velice omezené a byly výhradně orientovány na zobrazovací protokoly jako 3270. Ten řídil, jakým způsobem měly být znaky reprezentovány na obrazovce. Všechna vylepšení se týkala mainframů a některé z těchto inovací, zejména transakční zpracování a vývoj vysoko-úrovňových programovacích jazyků, významně obohatily průmysl, jakmile byly nasazeny.

Decentralizace

Příchod personálních počítačů s jejich možnostmi spouštět aplikace lokálně, převrátil uvedený model. Většina inovací se posunula od mainframe systémů směrem dolů ke klientským PC systémům. Personální počítače, zejména díky operačním systémům MS-DOS a Windows, se staly cílem zájmu softwarového průmyslu a umožnily vývoj bohatých klientských aplikací. Během této éry servery fungovaly jako poněkud chytřejší vrátní obsluhující a organizující soubory požadované klientskými aplikacemi.

Uzavření kruhu

S příchodem Internetu, World Wide Web a Web prohlížečů se situace obrátila ještě jednou a kyvadlo se zhouplo zpět směrem k více centralizovanému pohledu.



Obrázek 1. První generace modelu internetového: Webové portály přenášejí HTML do prohlížečů

Internet udělal zásadní převrat ve způsobu, jakým uživatelé pracují s aplikacemi a významně rozšířil použití osobních počítačů. Před deseti léty mělo mnoho z nás typicky k dispozici desítku aplikací. V současnosti si můžete sednou k prohlížeči, zadat URL a získat přístup k aplikacím a informacím kdekoli na světě.

Problém první generace internetového modelu spočívá v tom, že vám neumožňuje uzpůsobovat přijímané informace na klientské straně nebo provádět nějaké významnější nebo logické lokální zpracování. Jsme také omezeni k zadávání vstupu jednoduchou formou z klávesnice a významně limitování ve výběru klientských zařízení, kterými se lze k informacím dostat. V principu to je způsobeno omezenými možnostmi prezentačního protokolu HTML (Hyper Text Makup Languge).

Dnešní Internet je z převážné většiny pouhým světem prohlížečů, kde tyto nástroje poskytují pouze úzký náhled do oceánu dostupných informací. Jsme extrémně omezeni ve svých možnostech manipulace, modifikace nebo přidávání dalších hodnot k informacím, které získáváme.

Současný model interakce člověk-počítač je také zřetelně omezen a zahrnuje dva zcela oddělené světy. První, v němž si informace prohlížíme a druhý, v němž získané informace zpracováváme. Jako příklad si vezměte lištu aplikací dole na vaší obrazovce. Pravděpodobně máte neustále spuštěno několik instancí prohlížeče a pár dalších aplikací s nimiž vykonáváte svou práci. Je to příklad jednoho světa prohlížení a jiného světa zpracování informací.

Prohlížeč samotný je jen trochu více než grafická verze terminálu 3270 a jako terminál poskytuje jen málo v oblasti inteligentního klientského zpracování informace. Výkon potřebný pro zpracování je lokalizován na serveru, jenž generuje statický obraz výstupu ve formátu HTML a zasílá jej klientovi. Pokaždé, když potřebujete něco udělat, jste nuceni zaslat nový dotaz zpátky na server a požádat o další zpracování. Ve světě výkonných a levných klientských zařízení se klientskými kapacitami naprosto plýtvá.

Omezená interakce

Současný centralizovaný model orientovaný na prohlížeče je sice interaktivní, ale pouze ohraničeným způsobem. Uživatel musí sám započít interakci, např. zadáním URL nebo klikem na odkaz. Pro Internet je velice složité dělat věci automaticky ve jménu uživatele.

Dalším základním omezením dnešního centralizovaného modelu je neschopnost vzájemné komunikace jednotlivých serverů. Máte-li potřebu, je velice jednoduché přistoupit k libovolnému web portálu. Když však vyžadujete interakci libovolného

web serveru s jiným tak, aby poskytoval agregované nebo sloučené služby, vyžaduje to enormní množství nákladného specifického vývoje. V současnosti existují milióny webových portálů, ale všechny jsou jistými osamělými ostrovy. Je velice obtížné se současnými technologiemi dosáhnout, aby tyto portály navzájem reagovaly a spolupracovaly.

Peer to Peer

Pro jisté typy scénářů, kdy klienti přímo komunikují s jinými klienty a obcházejí server, poskytuje peer-to-peer model velký potenciál a přináší užitečná řešení. Zatím co centrální servery jsou potřebné k provedení jistých úkolů v jistých oblastech problémů, v ostatních případech nejsou třeba. Instant messaging, jaký nabízí například MSN®, je nejlepším a nejúspěšnějším příkladem řešení, které je postaveno na peer-to-peer modelu.

Zatímco mnoho nových a úspěšných aplikací poskytujících rozsáhlé komunikační prostředí je postaveno na peer-to-peer modelu, není zmíněný model vhodný pro všechny scénáře. Například pokud dva lidé potřebují spolupracovat asynchronně, je nutné nasadit server. Obdobně je server potřebný v případě, když aplikace požaduje, aby byl klient autentikován z důvodu bezpečnosti. Někdy budou také vaše klientská zařízení velmi jednoduchá a nebudou mít dostatek výkonu ke zpracování informací, které požadujeme. Další scénář, vyžadující přítomnost serveru pro vykonání náročných výpočtů a konverzních operací, je převod výsledků do formátu vhodných pro klientské zařízení.

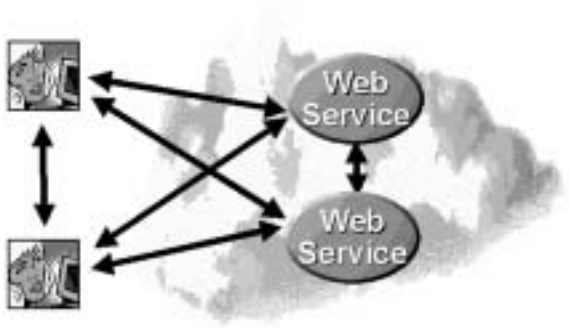
I přes svou důležitost poskytuje peer-to-peer model pouze částečné řešení.

Model webových služeb

V současné době lidé začínají pracovat s novou třídou zařízení, např. bezdrátová zařízení a zařízení připojená do širokopásmové televizní sítě. Skutečnost způsobuje významnou výzvu v oblasti vývoje serverově orientovaných aplikací. Pro vyřešení tohoto problému je velmi často budována množina souběžné infrastruktury, kdy jeden web portál je speciálně určen k poskytování informací pro personální počítače, další poskytuje obsah pro bezdrátová zařízení a třetí dodává obsah televiznímu přístroji. Uvedený typ redundance je zjevně neefektivní, drahý a vyžaduje alternativní přístup.

Dilema, kterému musí v současnosti čelit vývojáři, spočívá v tom, že musí vytvářet řešení uspokojující rozmanitý výběr klientských zařízení, lišících se rozlišením obrazovky a jinými faktory. A co více. Prezentované informace na těchto zařízeních musí být přizpůsobeny a často kombinovány z různých datových zdrojů poskytovaných širokou množinou různých serverů. U serveru je evidentně terminálový model pro realizaci klientského přístupu nebo čistého sdílení souborů naprosto neadekvátní. Řešení vyžaduje intenzivní výměnu informací a to v obou směrech.

Webové služby nabízejí takový přístup. Webové služby jsou softwarové komponenty nebo ostrůvky funkcionality, programově přístupné po Internetu za použití otevřených internetových protokolů včetně HTTP (Hyper Text Transfer Protocol) na úrovni transportu, a XML (eXtensible Markup Language) pro bohatou a pružnou reprezentaci dat. Jelikož je interakce mezi webovou službou a jejím klientem plně řízena otevřenými internetovými protokoly, jakékoli zařízení nebo platforma, která podporuje tyto standardy může hostovat nebo konzumovat Webové služby. Programový model webových služeb je plně platformově nezávislý a je ideální pro heterogenní podstatu Internetu.



Obrázek 2 Model webových služeb (software-to-software) v Microsoft .NET

Možnosti pro použití webových služeb jsou neomezené. Servery mohou velice jednoduše spolupracovat s jinými servery, mohou komunikovat s klienty a klienti mohou komunikovat navzájem. Například, když vejdete do jednací místnosti, máte v rukou klientské zařízení a v místnosti jsou další lidé s podobnými přístroji, byli by jste rádi, kdyby jste si mohli navzájem vyzvednout, předávat a vyměňovat informace. V tomto případě řešení vyžaduje klient-klient, klient-server a server-server elementy. Ty však budou postaveny na jednom standardu za použití vlastností a inteligence vestavěné do jednotlivých zařízení.

Potenciál pro inovace

Ve složitých prostředích již nadále nenahlížíme na informace z úzkého úhlu. Navštívit webový portál, aby jste získali informace o akciích nebo portfoliu, vyžaduje mnohem více než pouhou statickou stránku zobrazenou ve vašem prohlížeči. Chcete, aby informace byly předány přímo do aplikací jako je Microsoft Excel tak, aby jste s nimi mohli začít pracovat, přidávat nová data a provádět jejich inteligentní zpracování.

Tento přístup vyžaduje chytré klienty, např. klienty, kteří mohou pracovat v odpojeném režimu, provádět lokální zpracování a být nezávislými na serveru při vykonávání jednotlivých operací.

Vedle interaktivního přístupu, který je v současnosti běžnou věcí, uvidíme na Internetu čím dál tím častěji navzájem komunikující aplikace bez intervence člověka. Uvedení programového přístupu, jako je tento, je neuvěřitelnou silou a má dostatek potenciálu k transformaci služeb poskytovaných současným Internetem.
Jakmile se stane programový přístup realitou, mohou být milióny osamělých ostrůvků v Internetu navzájem jednoduchým způsobem propojovány a mohou se stát množinou do sebe zapadajících stavebních kamenů.

Podnícení kreativity

Díky otevřeným a platformově nezávislým standardům, na nichž jsou webové služby postaveny, není model web XML služeb spojen s konkrétní hardwarovou architekturou. Webové služby budou poskytovat infrastrukturu pro novou generaci Internetových aplikací, které transformují vaše zkušenosti s Internetem. I přesto, že obsahují peer-to-peer model na jedné straně, servery budou sehrávat významnou roli na straně druhé. Budete schopni přistupovat k personifikovaným a aktuálním informacím a informace vám budou prezentovány v jednom unifikovaném pohledu, navzdory faktu, že informace mohou pocházet z různých webových portálů a aplikací.

Platí to bez ohledu na typ klientského zařízení, které pro přístup k informacím používáte. Uživatelské rozhraní je zcela nezávislé na zpracování probíhající na webovém serveru. Když například chcete napsat webovou aplikaci, která obsluhuje dvacet typů telefonů s různým rozlišením displeje, rozložením kláves atd., serverová aplikace generuje XML s mapováním na konkrétní typ displeje telefonu a realizuje velmi jednoduchou logikou XML transformace, která se spustí buď na serveru nebo na klientovi v závislosti na jeho schopnostech a výkonu.

Jedním z nejvíce vzrušujících aspektů posunu k architektuře software-to-software je, že model webových služeb umožňuje neustálé přizpůsobování informace klientským potřebám a jejich upravení do formátu pro zařízení, které klient v aktuální chvíli využívá. Pokud například používáte nový typ klientského zařízení, po úspěšné autentikaci si zařízení samo nalezne odpovídající množinu webových služeb, které jste si vybrali pro správu informací a potřebná data nahraje. Nebudete muset data mezi aplikacemi manuálně replikovat.

Tento model také podporuje inteligentní informační agenty, neboli webové služby obsahující pravidla definující typy informací, na které by jste chtěli být upozorňováni. Mohou také cyklicky zjišťovat změny v informacích, které vás zajímají, např. nové produkty nebo změny v cenách a na ty vás upozorňovat. Uvedený přístup vám dává k dispozici možnost řídit sdílení informací a jak si organizovat svůj čas.

Pro podporu nového aplikačního modelu budeme potřebovat novou množinu kancelářského software. Bude vyžadovat bohatou podporu integrace s XML tak, aby využíval webové služby a aby dosáhnul na servery způsobem, jaký nebyl dříve možný. Jakmile se nový model začne soustřeďovat na používání vysokoúrovňových protokolů, umožní to neomezenou inovaci jak klientů tak serverů, v hardwarové i softwarové oblasti. Je důležité, že pokrok nastane v obou oblastech a tak žádná z nich nezbrzdí druhou.

TECHNOLOGIE SERVERŮ

Důležitý je taktéž pokrok v oblasti serverů. Internet je dnes plný serverů, které jsou příliš pomalé a havarují, protože nemají dostatek redundantních součástí. Windows 2000 byl prvním Microsoft operačním systémem, který obsahoval softwarovou technologii škálování, umožňující spojení několika fyzických serverů a jejich prezentaci jako jednoho logického celku. Softwarové škálování je základním nepostradatelným postupem jak pro škálovatelnost, tak pro zvýšení spolehlivosti.

Existuje stále dost práce na zjednodušení řízení těchto systémů a IT průmysl se na ni orientuje. Řešení založená na službách slibují přínos nižších nákladů a při použití nástrojů, které nabízí, budou schopny jednoduššího rozšíření s možností nasazení aplikací v mnohem kratším čase. Napomohou tak vylepšit vaši obchodní hbitost a reakce schopnost.

Současný rok lze také vnímat jako významný architektonický milník v oblasti serverů. Např. s příchodem 64-bitové technologie se začaly na trhu objevovat první systémy založené na Intel technologii InfiniBand. Zatímco v minulých letech společnosti jako Unisys přesunuly zákazníky k operačnímu systému Windows 2000, běžícím na 32-bitovém procesoru, tento rok postupně přecházejí na 64-bitové procesory.

KLIENTSKÉ SOFTWAROVÉ TECHNOLOGIE

Pokrok v klientském programovém vybavení je stejně důležitý jako pokrok u serverů. Klientský software bude například nabízet technologie rozpoznávání řeči.

Windows XP, které byly uvolněny na podzim roku 2001 jsou dalším krokem vpřed a obsahují mnoho nových vlastností. Jsou například schopny přejít ze standby módu již za několik sekund. Tato samotná vlastnost mění způsob, jakým budou lidé přistupovat k personálním počítačům.

Významně se také rozvíjí software pro online připojení. Software již není nadále založen čistě na HTML. Microsoft společně s MSN® v současné době uvolnil MSN Explorer, který je prohlížečem integrujícím mnoho vlastností a služeb a demonstruje směr, ve kterém půjde uživatelské rozhraní založené na XML. Umožňuje vám např. vytvářet přizpůsobitelné uživatelské rozhraní pomocí připojení a přizpůsobení již připravených komponent uživatelského rozhraní.

Časem vám pokrok v oblasti klientského software umožní rychlé a úsporné získání přizpůsobených dat z odlišných datových zdrojů a provádět inteligentní a automatické prohledávání jak v lokálních, tak ve vzdálených zdrojích. Nahradí se tím zdlouhavé a časově náročné manuální prohlížení.

TECHNOLOGIE PRO KLIENTSKÝ HARDWARE

Existuje mnoho kategorií klientského hardware. Personální počítače jsou v současnosti tou nejvýznamnější. Růst prodeje personálních počítačů v posledním roce je ve skutečnosti větší než součet u všech ostatních typů klientů. Bez obav však lze říci, že i ostatní typy klientů zvýší svou popularitu a v jejím důsledku Microsoft investuje do přizpůsobování software na různé klientské platformy pravděpodobně mnohem více, než ostatní společnosti.

Obrázek 3 Několik dobře použitelných zařízení

Personální počítače

Personální počítače samotné se stále rychle vyvíjejí. Zejména přenosné počítače urazily během krátké doby dlouhou cestu a jejich ceny se výrazně snížily. Ve standardu jsou často zabudovány mikrofony a reproduktory a k dispozici je velká spousta levných USB periferií. Rozlišení obrazovky se zvýšilo a pokrok jako ClearType technologie, která byla přidána do Windows, zvyšuje použitelnost různých zařízení.

PDA

Personal Data Assistant (PDA), jakým je například Pocket PC na platformě Windows, se rychle vyvíjí. Poptávka po těchto typech zařízení je neobyčejná. Společnosti jako Compaq a HP výrazně zvýšily objem své výroby a stále nejsou schopny uspokojit aktuální potřeby.

Současná PDA obsahují barevné obrazovky, podporu pro streaming media a další pokročilé technologie. Vládnou dokonce mnohem větším výkonem, než měly personální počítače před deseti lety. I když se komunikační rychlosti zvyšují, limitujícím faktorem je velikost obrazovky.

Chytré telefony

Hodnota mobilních telefonů je dále umocňována chytrými variantami. Ty poskytují směs online a offline aplikací pro přístup k důležitým a časově kritickým informacím. Microsoft plánuje spojení toho nejlepšího z oblasti PDA a nejlepšího z mobilních telefonů při realizaci platformy, která poskytne lidem inteligentní konektivitu – ať hlasem, elektronickou poštou nebo jiným způsobem. Smart Phone, platforma firmy Microsoft, nabídne svým uživatelům jak existující osobní informace a elektronickou poštu, tak vysoce robustní možnosti prohlížení webových stránek. Platforma bude obsahovat webový prohlížeč s podporou HTML, WAP (WML) a XML formátů.

Xbox

Další oblastí, do které Microsoft silně investuje, je zábavné zařízení pod názvem Xbox, optimalizované pro připojení k televiznímu přijímači. Podrobnější informace o Xboxu můžete najít na adrese ***www.microsoft.com/xbox.***

eBook

Mnoho vydavatelů již dnes vydává elektronické knihy neboli eBooks specificky pro Microsoft Reader, software pro zobrazovaní elektronických knih využívající technologii ClearType pro zobrazení písma na displejích. Podrobnější informace o eBook naleznete na adrese ***www.microsoft.com/reader/default.asp.***

Tablet PC

Prototypy těchto zařízení již byly demonstrovány při mnoha různých událostech. Microsoft při vývoji Tablet PC úzce spolupracuje s výrobci procesorů a disků, a dále se specialisty na technologii rozpoznávání písma.

Tablet PC podporuje tzv. přímou manipulaci, která vám dovoluje ukázat na položky na povrchu tabletu za účelem jejich označení, výběru a posunutí. Přímá manipulace nebude pouze omezena na zařízení typu Tablet PC, nýbrž bude podporována většími CD přehrávači, které budete mít na svém stole nebo nástěnkami na stěnách vašich kanceláří.

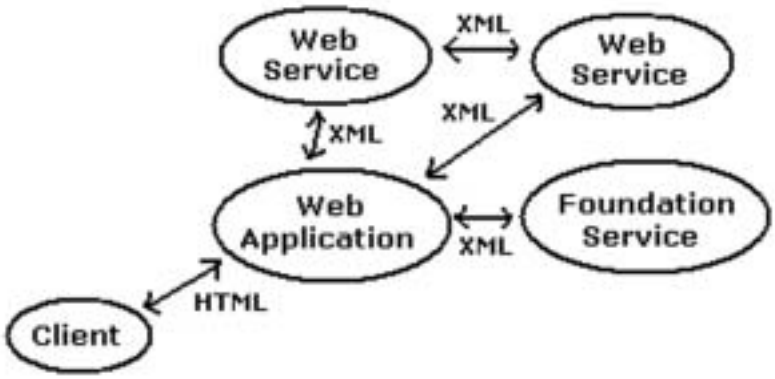
Přímá manipulace společně s rozpoznáváním řeči se stanou klíčovými novými elementy platformy Windows připravené pro novou generaci aplikací.

Obrázek 4 Tablet PC

Uvedený seznam nám přestavil některá z nových vzrušujících zařízení, která jsou dostupná již dnes nebo ve velmi blízké budoucnosti. Mezi neustále rostoucím množstvím klientských zařízení však zůstanou personální počítače klíčovým hráčem. Všechna tato zařízení budou hrát doplňkovou úlohu a na základě podpory XML budou schopna různými způsoby spolupracovat a vzájemně reagovat.

MICROSOFT .NET VIZE

Microsoft .NET vize a mise, ve které se Microsoft angažuje, je zaměřena na podporu vývojářů aplikací při realizaci internetových distribuovaných aplikací, analogických současné Windows platformě. Filozofie spočívá v umožnění vývoje sofistikovaných a vysoce integrovaných aplikací zahrnující XML interakci mezi mnoha webovými portály a webovými službami.



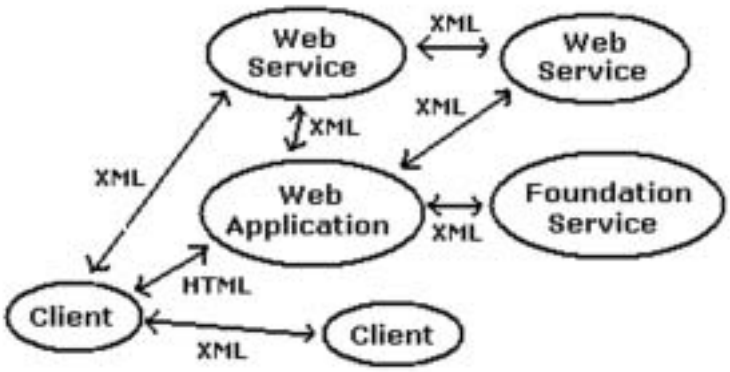
Obrázek 5 XML umožňuje komunikaci mezi servery a vývoj Základních služeb.

Tato platforma je od základu založena na XML společně s příbuznou rodinou standardů. Schopnost XML přenášet samo-popisná data na platformově neutrálním základě dává příslib pro vyřešení tohoto základního softwarového požadavku a nabízí pružnost a konzistentnost, které dosud nebylo dosaženo.

.NET přístup je soustředěn kolem tzv. Základních služeb, také známých jako Základní stavební služby. Jde v postatě o internetové komponenty poskytující opakovaně použitelné programové funkcionální elementy. Microsoft .NET se zaměřuje na vybavení programátorů sadou základních stavebních služeb, které mohou být kombinovány s ostatními základními službami společně se zákaznickou aplikační logikou, aby významně napomohl vývoji distribuovaných aplikací.

Ukázkovým příkladem stavebních služeb je Microsoft Passport. Passport je dnes téměř šestým nejnavštěvovanější webovým portálem na světě, avšak jen velmi málo lidí někdy zadalo přímo do prohlížeče adresu http://www.passport.com. To přesně ilustruje klíčovou myšlenku. Passport poskytuje službu, která může být zakomponována do jiných web aplikací na jiných portálech. Umožní se tak webovým portálům out-sourcing autentikačního procesu. Passport udržuje databázi uživatelů a řídí, jestli má být konkrétní uživatel do systému vpuštěn či nikoli.

Vzrůstající množství dat bude v budoucnu přesouváno na klienty pomocí XML. Data přijímaná klienty již nadále nebudou statickým obrazem toho, co se děje na serveru, ale přijetí čistých dat umožní tlustým klientům provádět jejich zpracování, manipulovat s daty a přidávat k nim další informace.



Obrázek 6 Webové služby a tlustí klienti

Vedle komunikace se servery podporuje tento přístup i peer-to-peer model, kde klienti mohou přímo komunikovat s ostatními klienty. Technika webových služeb ústí v model distribuovaného zpracování, který se nesnaží protlačit každou operaci přes server a snižuje tím potenciální riziko úzkých profilů.

Další výhoda modelu založeného na službách spočívá ve schopnosti klientů přijmout nejnovější verze jednotlivých kusů programového vybavení. To je v přímém kontrastu s dnešním modelem u něhož si musíte pořídit nové CD pro instalaci update. Čelíte-li otázce podpory nebo například nevíte, jak přesně použít jistý software, můžete využít sítě k získání rady a zjištění, zda se někdo jiný neocitl v podobné situaci. Bude-li to nutné, budete se schopni přímo spojit formou peer-to-peer a získat požadovanou podporu a radu. V budoucnosti bude dostupné ohromné množství software za základě měsíčního předplatného se zajištěním podpory, jako součástí uzavřené smlouvy.

Vývojové nástroje

Pro zjednodušení vývoje aplikací s webovými službami a aby bylo zajištěno, že webové služby se opravdu rozšíří, je nutné mít k dispozici nové vývojové nástroje pracující na vyšší úrovni než nástroje v současné době. Jako analogii si lze uvést dobu, kdy bylo uvedeno grafické uživatelské rozhraní. Trvalo dlouhou dobu, než bylo vyvinuto dostatečné množství aplikací. Stalo se tak proto, že na počátku nebyly k dispozici potřebné vývojové nástroje a všichni byli přinuceni pracovat na velmi nízké úrovni, jako bylo programování smyček zpráv a pod.

Visual Studio.NET se stává hlavním vývojovým nástrojem pro Microsoft .NET Framework a umožňuje vývojářům rychle a jednoduše vyvíjet webové služby a internetové aplikace další generace za použití vysoko-úrovňových nástrojů. Určuje cestu pokroku vývojářům v jazycích C a C++, vývojářům používající jazyk Visual Basic i Java programátorům.

Jazyky a technologie, které Microsoft vyvíjí, nejznatelněji nový jazyk C# (vyslovuje se sí-šarp) a Common Language Runtime (CLR) poskytující prostředí pro spuštění .NET aplikací, byly předány organizaci ECMA (European Computer Manufactures Association) ke standardizačnímu řízení. To je velmi důležité, protože standardizační organizace může technologie přijmout, uvést ji jako novinku a přijímat různé inovace od různých společností, včetně konkurentů firmy Microsoft.
C# a Common Language Runtime existují pro každé zařízení a pro každý operační systém, nikoli jen pro platformu Windows. XML je klíčovým elementem, protože umožňuje různým platformám vzájemnou integraci. Každá platforma podporující exekuční prostředí může jednoduše soutěžit se svou implementací a přidanou množinou služeb, která překračuje základní funkcionalitu.

PŘÍLOHA

ODKAZY NA INFORMAČNÍ ZDROJE

Následující seznam není úplný a vyčerpávající přehled článků, ale nasměruje vás na ty nejzajímavější a nejsrozumitelnější články o .NET.

Vize

http://www.microsoft.com/business/vision/netwhitepaper.asp
http://www.microsoft.com/presspass/futures/2000/sept00/09-26sentprise.asp
http://www.microsoft.com/business/vision/dafault.asp
http://www.microsoft.com/presspass/futures/2000/sept00/09/06uddi.asp
http://www.key3media.com/comdex/fall2000/daily/webcast/index.html

Exekutivní pohledy

http://www.microsoft.com/business/vision/gates.asp
http://www.microsoft.com/business/vison/ballmer.asp
http://www.microsoft.com/business/vision/muglia.asp

Obecné portály s informacemi

http://www.microsoft.com/net
http://msdn.microsoft.com/net
http://www.GetDotNet.com
http://www.asp.net
http://commnet.pdc.mscompevents.com
http://www.microsoft.com/server/net/default.htm
http://www.ibuyspy.com

Kapitola 2:

Co je to Microsoft .NET?

Abstrakt

Tento kapitola se zabývá otázkou, co znamená .NET vize pro vás a pro budoucnost softwarových aplikací. Popisuje .NET vizi a vysvětluje konkurenční výhody, které přinese v oblasti obchodu.

ÚVOD

Microsoft .NET urychlí posun průmyslu k distribuovanému výpočetnímu modelu založenému na Internetu. Microsoft .NET se stává, díky internetovým standardům, platformou webových XML služeb pro zprostředkování vzájemné spolupráce aplikací, služeb a zařízení a umožňuje jim přístup a zpracování informací kdykoli, kdekoli a z jakéhokoli zařízení. Poskytuje jednoduchý vývoj, bohatou zkušenost uživatelů a nabízí nebývalé možnosti pro rozvoj obchodu.

Microsoft .NET je:

- .NET platforma s následujícími komponentami:
  - .NET Framework a vývojové nástroje Visual Studio.NET
  - .NET základní stavební služby
  - .NET klientský software pro chytrá zařízení
  - Microsoft .NET Enterprise servery
- Zkušenost .NET uživatelů postavená na této platformě.

ZÁSADNÍ PRINCIPY MICROSOFT .NET

Tato část prozkoumává některé zásadní změny, které nastávají uvnitř softwarového průmyslu. Abychom byli schopni tyto změny přijmout a využít plně jejich výhod, potřebujeme nové přístupy, nové programové modely a nové architektury.

Distribuované výpočty založené na internetových standardech

Softwarový průmysl se velmi rychle rozvíjí a přijímá Internet jako výkonnou a bohatou infrastrukturu, na níž lze vyvíjet distribuované aplikace ohromného rozsahu. Společnosti stále větší měrou vytvářejí řešení pro integraci a konsolidaci struktur systémů založených na LAN sítích nebo samostatně stojích mainframech a formuje z nich výkonné distribuované aplikace. Různé vrstvy takovýchto aplikací mohou být buď připojené nebo odpojené, lokální nebo vzdálené, na stejném počítači nebo distribuované přes Internet. Tím jsou dány k dispozici možnosti inteligentního návrhu s mnohem efektivnějším zpracováním. Bohatá komunikační infrastruktura poskytovaná Internetem nám umožňuje využití výpočetního výkonu tam, kde to dává smysl, ať jde o výkonné servery nebo o aplikační vrstvu na tučných klientech.

Jelikož nás čeká mnohem delší historie Internetu než je za námi, je logické vytvořit novou platformu. Platforma, která využívá neomezené možnosti Internetu, kde lze aplikace vytvářet nezávisle nebo použít existující služeb formou jednotného a otevřeného modelu.

Moorův zákon a komunikace

V roce 1965 učinil Gordon Moore památné pozorování. Uvědomil si, že zde existoval evidentní trend v růstu výkonnosti mikročipů. Kapacita každého nového mikročipu, vyprodukovaného v období od 18 do 24 měsíců od jeho předchůdce, se zvýšila přibližně dvakrát. Moorovo pozorování, známé jako Moorův zákon, popsalo trend, který pokračuje a je neustále nevšedně přesný.
Obdobný trend nastává i v komunikačních technologiích. Cena přenosového pásma klesá přibližně na polovinu každé čtyři měsíce. Microsoft se zaměřil na to, aby jeho budoucí architektury byly schopny těžit z obou těchto faktorů.

Klíčem je integrace

Směr pozornosti při realizaci řešení se přesunul od jednoduchých aplikací ke konfiguracím zahrnujícím integrovaná zařízení, služby a aplikace. Řešení, která vytváříte, již nejsou nadále postavena na jediné aplikaci běžící na jednom počítači, nýbrž se sestávají z mnoha zařízení, mnoha aplikací a mnoha služeb, které mohou být distribuovány po celém světě. Základní nutností je platforma, která dramaticky zjednodušuje složení řešení.

Software a služby

Posuv směrem k softwaru postavenému na službách je v kontrastu se starým světem distribuce programového vybavení na CD discích vyžadujícího cyklické upgrady. Služba vám poskytuje přímé spojení k poskytovateli a nabízí nepřerušený vztah a nepřerušenou dodávku nových verzí přímo od poskytovatele.

Nová generace uživatelských zkušeností

Existuje zde obrovská možnost jak vylepšit způsob komunikace s aplikací. Zatímco grafické uživatelské prostředí představovalo obrovský krok vpřed ve zvýšení uživatelských zkušeností, nyní nastal čas, abychom připravili ještě lepší uživatelská rozhraní. Rozhraní, která jsou adaptivní, inteligentní, založena na identitě uživatele umožňují personalizovat dodávaný obsah. Navíc technologie, jako je rozpoznávání řeči a rozpoznávání písma, se rychle rozvíjejí a v budoucnosti úplně přetvoří způsob, jakým budeme s aplikacemi komunikovat.

Předání řízení uživateli

Klíčovým prvkem technologie a aplikací, se kterými pracujete, by jste měli být právě vy. Aplikace by měli pracovat ve váš prospěch a pod vaším velením. Je to celkem odlišný přístup od současné situace, kdy se musíte částečně přizpůsobovat, konfigurovat své aplikace a stát se tak lidským mostem mezi různými zařízeními, aplikacemi a webovými portály, se kterými komunikujete.

.NET VIZE

Před pouhými 20 lety byl svět stále v éře mainframů. Pouze pár lidí mělo přístup k počítačům a to pouze formou nejbližšího centra výpočetní techniky. Personální počítače a grafické uživatelské rozhraní vše změnilo a přiblížilo výpočetní techniku miliónům lidí, transformujíc počítače na skutečný masový produkt. Podniky si uvědomily, že sítě personálních počítačů a PC serverů mohou změnit způsob, jakým dělají obchod. Mezi běžnými uživateli se personální počítač velmi rychle etabloval jako nové médium pro domácí zábavu. Internet uskutečnil revoluci ve způsobu, jakým spolu lidé komunikují, založil nový a bohatý zdroj informací a zábavy a otevřel svět elektronickému obchodu. V současné době využívá Web na celém světě téměř 300 miliónů lidí. Podle Intenational Data Corp. prošlo obchodními transakcemi na Internetu v loňském roce více než čtvrt biliónu dolarů.

Všechny tyto divy mají stále dostatek prostoru k vylepšování. Soudobý Internet silně odráží starý mainframe model. I přes štědrou šířku komunikačního pásma, jsou informace stále uzavřeny v centrálních databázích s řízeným přístupem. Uživatelé se musí spoléhat na webový server při vykonání každé operace, stejně jako tomu bylo ve starém modelu se sdílením strojového času. Webové portály jsou izolované ostrůvky a nejsou schopny ve prospěch klienta spolu navzájem jakýmkoli způsobem komunikovat. Současný Web je jen o něco více než jednoduché poskytování individuálních stránek individuálním uživatelům - stránky, které nejčastěji představují data v HTML „obrazu“, nikoli data samotná (v současnosti je velice obtížné pro většinu web portálů poskytovat obojí). A oslavovaný prohlížeč je v mnoha ohledech přihlouplý terminál určený pouze ke čtení. Velice jednoduše si můžete informace prohlížet, ale je obtížné je editovat, analyzovat nebo s nimi manipulovat (to vše jsou operace, které potřebují dělat lidé pracující s informacemi). Personalizace spočívá v opakovaném zadávání a ztrátě kontroly nad osobními informacemi, které poskytnete každému navštívenému portálu. Vy se musíte přizpůsobovat technologii místo toho, aby se technologie přizpůsobovala vám.

Tyto problémy se znásobí, pokud používáte více než jedno PC nebo mobilní zařízení. Při online přístupu k vašim informacím (e-mail, offline soubory a další data) musíte bojovat s mnoha rozhraními (často nekompatibilními), měnící se úrovní přístupu k datům a pouze občasnou synchronizací všech informací, které potřebujete (např. když vaše zařízení fyzicky připojíte k personálnímu počítači). Online data jsou prezentována neúplným a předdefinovaným formátem, významně omezujícím jejich použití. Koncepce přizpůsobitelného „informačního prostoru“, který se chová přesně podle vašich potřeb, je stále pouhým snem.

Aktuálně dostupné nástroje sloužící webovým vývojářům pro vytváření, testování a nasazení webových portálů jsou zoufale nedostatečné. Mnohé se více orientují na vytváření atraktivních než použitelných webových portálů. Žádný z nich neoslovuje úplný životní cyklus software, od návrhu přes vývoj, nasazení až po údržbu, způsobem, který by byl konzistentní a úsporný. Žádný současný systém neumožňuje vývojářům napsat kód pro PC a nasadit jej na velké množství ostatních zařízení.

Uživatelé společností čelí navíc další výzvě. Poté, co byly zavedeny počítačové farmy, které se sestávají z malých serverů poskytujících celkově spolehlivé výpočetní prostředí odstraněním jediného místa vniku poruch, stala se správa těchto systémů mnohem obtížnější. Měření výkonnosti, plánování kapacity a řízení operací jsou v dnešním světě mnohavrstvých a mnohoúčelových webových portálů výzvou. Nové systémy elektronického obchodu jen velmi obtížně spolupracují se staršími obchodními systémy. Postavení systémů, které bezpečně překonávají firewall prvky tak, aby se uživatelé a partneři mohli inteligentně angažovat v obchodě, je natolik obtížné, že mnoho firem se uchýlilo k duplicitním systémům.

Uvádíme Microsoft .NET

Microsoft vytváří novou pokročilou generaci softwaru, který spojuje výpočty a komunikaci zcela novým revolučním způsobem a nabízí vývojářům nástroje, které potřebují pro transformaci Webu a všech ostatních aspektů týkajících se zkušeností s výpočetní technikou. Tato iniciativa se jmenuje Microsoft .NET a poprvé umožňuje vývojářům, podnikům a zákazníkům vyžít technologii podle svých představ. Microsoft .NET umožní vytvoření skutečných distribuovaných webových služeb, které budou integrovat a spolupracovat se širokým rozsahem doplňkových služeb, za účelem obsluhy zákazníků. Microsoft .NET bude řídit Novou Generaci Internetu. Iniciativa opravdu zpřístupní informace kdykoli, kdekoli a na jakémkoli zařízení.

Základní myšlenou stojící za Microsoft .NET je změna orientace od individuálních webových portálů nebo zařízení spojených přes Internet ke struktuře počítačů, zařízení a služeb, které spolupracují a poskytují širší a bohatší řešení. Lidé budou mít možnost ovlivnit jak a kdy jim jsou informace předávány. Počítače, zařízení a služby budou schopny navzájem spolupracovat a poskytovat bohatší služby, místo toho, aby byly izolovanými ostrůvky, kde integračním článkem je uživatel. Podniky budou schopny poskytovat své produkty a služby takovým způsobem, že je budou klienti bezproblémově začlenit do své vlastní elektronické struktury. Je to vize, která rozšiřuje individuální možnosti poprvé poskytnuté personálními počítači v roce 1980.

Microsoft .NET pomůže řídit transformaci Internetu, která se projeví v prezentacích založených na HTML, rozšířených o programovatelné XML informace. XML je široce podporovaným průmyslovým standardem definovaným World Wide Web konsorciem, stejnou organizací, která stanovila standardy pro web prohlížeče. Byl vybudován se silnou účastí Microsoftu, ale nejde o proprietární Microsoft technologii. XML dává k dispozici prostředky pro oddělení vlastních dat od prezentačního pohledu na tato data. Jde o klíčovou záležitost pro Novou Generaci Internetu a umožňuje otevřít informace pro následovnou organizaci, programování a editování. Dále umožní mnohem užitečnějším způsobem distribuovat dat na různá digitální zařízení a webovým portálům zprostředkuje strukturu pro spolupráci s webovými službami, které budou schopny vzájemně komunikovat.

Microsoft .NET posune výpočty a komunikaci mnohem dále za jednosměrný Web, směrem k bohatšímu, kooperujícímu a interaktivnímu prostředí poháněného novým pokročilým software. Microsoft .NET využije strukturu aplikací, služeb a zařízení k tomu, aby přispěl k nové zkušenosti uživatelů. Zkušenost, která se sama neustále a automaticky přizpůsobuje vaši potřebám, potřebám vaší rodiny, domova a obchodu. Znamená to zcela novou generaci software, který bude fungovat jako integrovaná služba a pomůže vám řídit váš život a práci v Internetovém Věku.

Pro spotřebitele to znamená jednoduchost integrovaných služeb čítajících jednotné prohlížení web stránek, editaci a vytváření dokumentů, přístup ke všem souborům, práci a médiím jak online, tak offline, celkovou zkušenost se zařízeními, personalizaci na každém místě a nulovou administraci. Znamená to například, že libovolná změna ve vašich informacích provedená formou PC, handheldu nebo smart kreditní kartou – bude okamžitě a automaticky dostupná na libovolném jiném zařízení. Pro lidi pracující s informacemi a pro obchody to znamená sjednocené prohlížení, editaci a vytváření dokumentů, bohatou komunikaci při řízení, bezproblémovou integraci mobilních technologií, výkonný informační management a nástroje elektronického obchodu, které se budou transparentně pohybovat mezi interními a internetovými službami a budou podporovat novou éru dynamických obchodních vztahů.

Pro nezávislé vývojáře software to znamená možnost vytvářet nové pokročilé služby pro Internetový Věk. Služby, které jsou schopny automaticky přistupovat a těžit z lokálních nebo vzdálených informací, pracovat s libovolným zařízením nebo jazykem, bez nutnosti přepisovat kód pro každé prostředí. Vše na Internetu se stává potenciálním stavebním kamenem pro tuto novou generaci služeb, kdy každou aplikaci lze na Internetu vystavit jako službu.

Microsoft .NET vize znamená posílení spotřebitele, obchodu, vývojáře programového vybavení a celého průmyslu. Znamená to zpřístupnění plného potenciálu Internetu.

Další generace Internetu

Microsoft .NET je platformou pro vytváření nové generace distribuovaných aplikací založených na standardní kostře sestávající se z XML a internetových protokolů. V minulosti byly programové modely zaměřeny na jeden systém, a dokonce se snažily skrýt interakci s jinými systémy a předstírat, že jde o lokální interakce. Microsoft .NET je explicitně navržen za účelem integrace nebo orchestrace libovolných zdrojů na Internetu do formy jednoho řešení. V současné době je tento typ integrace extrémně složitý a drahý. Microsoft .NET jej učiní opravdovou skutečností pro veškerý vývoj programového vybavení.

Volně vázaný Microsoft .NET XML programový model zavádí koncepci vytváření webových XML služeb. Zatímco současné webové portály jsou řemeslnou ruční prací a nespolupracují bez dalšího vývoje s ostatními portály, programový model Microsoft .NET poskytuje skutečné mechanismy pro vytváření webových portálů nebo služeb tak, že budou schopny federovat a spolupracovat bez problému se všemi ostatními. Stejně jako zavedení zaměnitelných součástí odstartovalo průmyslovou revoluci, tak Microsoft .NET slibuje urychlení vývoje Nové Generace Internetu.

Nic z toho se nestane skutečností bez mnoha partnerů a miliónů nezávislých vývojářů i firemních vývojářů, kteří pomohli vybudovat dnešní počítačový průmysl. Jak řekl Alexander Graham Bell, „Velké objevy a zdokonalení vždy vyžadují mnoho názorů“. Když se stal na osobních počítačích populárním diskový operační systém (DOS), vytvořil nové generaci nezávislých vývojářů příležitost realizovat obchod s DOS aplikacemi. Operační systém Windows firmy Microsoft posunul tyto možnosti na ještě vyšší úroveň. Možnosti skýtané každému vývojáři ve formě Microsoft .NET jsou ještě větší.

Microsoft vytváří zcela novou množinu Microsoft .NET vývojových nástrojů speciálně navržených pro realizaci Webu, klientů, serverů a služeb. Tyto nástroje umožní vývojářům transformovat Web z dnešní statické podoby prezentace informací na Web s bohatými interaktivními službami. Průlomová nová generace sady nástrojů Microsoft Visual Studia automatizuje vývoj webových služeb formou rychlých drag-and-drop postupů navržených vývojovým týmem Visual Basicu. Tyto služby mohou být konzumovány na libovolné platformě, která rozumí XML. Visual Studio dokonce automaticky generuje XML kód.

Programový model Microsoft .NET poskytuje nezávislým vývojářům možnost koncentrovat méně energie na to jak, nebo kde aplikace běží, a raději se zaměřit na to, co aplikace dělá – na to, kde mohou přidat skutečnou hodnotu. Microsoft .NET reaguje na jedny z největších výzev před nimiž stojí vývojáři, zejména kompromis mezi funkcionalitou a schopností managementu. Posouvá tím poskytovatele aplikačních služeb (ASPs) a hostování aplikací na novou úroveň. Umožňuje to integraci a hostování aplikací společně s dalšími aplikacemi (ať hostovanými nebo nikoli), přizpůsobení těchto aplikací na zakázku, schopnost programovat proti těmto aplikacím a možnost spouštět aplikace v offline režimu.

Vývojáři budou navíc schopni ovlivnit a přizpůsobit celý rozsah základních Microsoft .NET stavebních služeb v rámci svých vlastních aplikací a služeb a snížit tak úsilí potřebné pro vývoj konkurenčních produktů. Základní Microsoft .NET stavební služby korespondují s oblastmi funkcionality, v nichž má Microsoft hluboké zkušenosti a může poskytnout hodnoty širokému rozsahu vývojářů. V mnoha případech Microsoft sjednocuje vývojářské stavební bloky uvnitř operačních systémů Windows s podobnými možnosti jaké jsou dnes dostupné v Internetu. To zjednodušuje produkci vysoce distribuovaných programových služeb, které běží na samostatných strojích, ve firemních datových centrech a po Internetu.

S možností registrace ke konzumaci základních Microsoft .NET služeb, se vývojáři mohou rozhodnout mezi variantami „napsat nebo koupit“ podle toho, jak chtějí nasadit své vývojové kapacity. Někteří se mohou rozhodnout pro vlastní vytvoření jednoduché základní funkcionality. Mnozí se pravděpodobně rozhodnou pro dobře provedené řešení se silnou podporou vývojových nástrojů, stejně jako mnoho vývojářů se rozhodlo nevytvářet si své vlastní ovladače tiskáren nebo grafický subsystém jako mají Windows, ale místo toho soustředilo své zdroje na odlišnou implementaci svých vlastních vysokoúrovňových produktů.

Nová generace produktů a služeb

V dlouhém horizontu bude mnoho aplikací dostupných nebo dodávaných jako předplacené služby na Internetu. Software jako služba umožňuje následující konkurenční výhody:

- lepší zákaznický servis
- transparentní instalaci, opravy a update
- těsnější zpětnou vazbu během životního cyklu aplikace
- rychlejší odezvu na softwarové problémy jako je antivirová ochrana.

Microsoft předpovídá, že většina jeho softwarových aplikací se časem převede na předplatné služby, přičemž bude udržovat a nabízet existující platformy a aplikace. Již od počátku bude Microsoft nabízet řadu .NET produktů a zkušeností, včetně následujících:

- Novou generaci desktopové Windows platformy, která podporuje produktivitu, kreativnost, administrovatelnost, zábavu a mnohé další a je navržena tak, aby umožnila uživatelům ovládat svůj digitální život. Díky úzké integraci se základní množinou .NET stavebních služeb nabízí integrovanou podporu pro digitální média, spolupráci a lze ji personifikovat. Může být také naprogramována pomocí .NET služeb, MSN.NET a bCentral®.
- MSN.NET – Formou kombinace obsahu a služeb přední služby MSN a nové platformy .NET, MSN.NET umožní svým uživatelům vytvoření jednotné digitální osobnosti a využití inteligentních služeb pro konzistentní, bezproblémový a bezpečný přístup k informacím, zábavě a lidem, na kterých jim záleží, v libovolné chvíli, místě a z libovolného zařízení. MSN.NET bude postaveno na nových integrovaných klientech, které se nacházejí v současné době v beta verzích.

- Personal Subscription Services – Microsoft vytvoří, vedle MSN.NET, sadu prémiových zákaznických služeb na platformě .NET, které budou postaveny nad existujícími zábavními, výukovými a produkčními produkty. Tyto služby poskytnou lidem výkonnost tradičních desktopových aplikací spolu s pružností, integrací a podporou celé funkčnosti rodiny .NET.
- Office.NET - Pokročilé komunikační a produkční nástroje, obsahující universální základní technologii kombinující nástroje pro komunikaci, prohlížení a tvorbu dokumentů do jednotného prostředí. Zprostředkuje uživatelům jednotným způsobem kombinovat a reagovat na informace. Univerzální kolaborační služby zajistí spolupráci mezi lidmi jak uvnitř, tak vně případných společností. Architektura založená na standardech, nasazení chytrých klientů a služeb poskytne bohatou funkcionalitu, výkonnost a automatické nasazení na libovolné zařízení. Microsoft také bude pokračovat s nabídkou podpory ostatních verzí Office bez podpory .NET služeb.
- Visual Studio.NET – programový model a nástroje založené na XML s plnou podporou MSDN® (Microsoft Developer Network) a .NET Enterprise Serverů. Nabízí jednoduchý vývoj vysoce distribuovaných a programovatelných služeb, které běží napříč jednotlivými počítači, datovými centry a přes Internet.
- bCentral ™ pro .NET – Velmi progresivní skupina předplacených služeb a nástrojů pro malé a rostoucí podniky. Obsahuje hostovaný management zpráv, elektronické pošty, pokročilé obchodní služby a nový systém správy zákazníků (CRM), formou služeb postavených na .NET platformě. Pokročilé služby obchodu a managementu vztahů se zákazníky umožní malým podnikům snadněji a lépe poskytovat online služby svým zákazníkům. Funkcionalita bude také obsahovat podporu pro rozsáhlé katalogy zboží a sledování interakce se zákazníky za účelem individuálního přístupu.

CO JE MICROSOFT .NET?

Microsoft .NET se zaměřuje na urychlení posunu IT průmyslu k internetovému distribuovanému výpočetnímu modelu.

Microsoft.NET je platformou webových XML služeb, založených na otevřených Internetových protokolech, poskytujících aplikacím, službám a zařízením schopnost vzájemně spolupracovat a umožnit jim přístup k informacím v libovolnou chvíli, libovolném místě a z libovolného zařízení. Poskytuje tak zjednodušený vývoj, bohatou zkušenost uživatelů a nebývalé příležitosti pro obchod.

Abychom lépe porozuměli tomu, co Microsoft .NET reprezentuje, pomůže nám rozdělení výše uvedené definice na jednotlivé části a její analýza z různých pohledů:

- „Microsoft .NET se zaměřuje na urychlení posunu IT průmyslu k internetovému distribuovanému výpočetnímu modelu.“:
  Dochází k tichému posunu k distribuovanému výpočetnímu modelu založenému na Internetu. Microsoft poskytuje nástroje a technologie pro distribuované výpočty již několik let. Ty však byly optimalizovány pro scénáře v Intranetových a LAN sítích. S rychlým rozvojem Internetu během několika posledních let se stávají vysoké přenosové rychlosti levnější a veřejně dostupné než kdykoli před tím. V kombinaci s Moorovým zákonem máte nyní nejlepší možnost vytvářet distribuované aplikace v prostředí Internetu jako komunikační infrastruktuře. Protože přenosové pásmo je levnější, můžete provádět zpracování dat v místě, které je nejvýhodnější a optimální. Co více, můžete plánovat nové scénáře, o kterých jste dříve nemohli ani uvažovat, jako je integrace obchodních procesů mezi partnerskými organizacemi pomocí webových XML služeb.
- „Založen na otevřených Internetových standardech“:
  .NET je postaven od úplného počátku na otevřených Internetových standardech jako jsou XML, HTTP, SOAP, UDDI a DISCO. Taktéž .NET CLR (Common Language Runtime) a jazyk C# byly předány ECMA (European Computer Manufacturer's Association) ke standardizaci. Microsoft je zavázán těmito otevřenými průmyslovými standardy. Výzvou pro Microsoft je vytvoření nejlepší implementace těchto standardů na trhu použitím svých produktů a nástrojů.
- „Microsoft .NET je platformou webových XML služeb“:
  Středem Microsoft .NET platformy jsou webové XML služby. Jde o softwarové komponenty programově přístupné přes Web, používající protokoly XML a HTTP (SOAP). K webovým službám lze přistupovat z různých klientských aplikací či zařízení a umožnit tak daleko lepší konektivitu jak pro uživatele, tak pro obchod.
- „Poskytujících aplikacím, službám a novým zařízením schopnost“:
  Platforma podporuje desktopové aplikace, centralizované služby, internetové aplikace, integrace obchodních procesů, bezdrátová zařízení, telefony a rozšiřitelnost na nová zařízení. Tyto schopnosti jsou možné díky společné technologie zabudované do .NET Enterprise serverů a Visual Studia.NET. Vedle desktopů, serverů a internetových klientů jsou podporovány nové bezdrátové a telefonní protokoly jako 802.11 a WAP (Wireless Application Protocol). Taktéž je podporováno mnoho dalších zařízení jako jsou Pocket PC, telefony a zařízení určená pro zábavu.
- „Vzájemně spolupracovat“:
  Integrace je jedním z klíčových cílů a dovoluje realizaci řešení v širším měřítku, která lze zpřístupnit mnoha klientům a zařízením, ať jsou lokální nebo dostupná přes Internet. XML a SOAP zajišťují integrační technologie pro bezproblémovou integraci.

- „Umožnit jim přístup a práci s informacemi“:
  .NET platforma zpřístupňuje informace a datové zdroje napříč systémy, klienty a zařízeními. Avšak přístup k informacím a datům není dostatečný, pokud nemohou být modifikována, zpracovávána a integrována. Platforma poskytuje přístup k datům jak pro čtení, tak pro zápis. Založením webových XML služeb dochází ke zpracování buď na klientské, nebo serverové straně a přináší rozsáhlé možnosti pro široký rozsah zařízení.
- „V libovolnou chvíli, libovolném místě a z libovolného zařízení“:
  Je zcela jedno, jestli jste na letišti pouze s mobilním telefonem v ruce. Pokud potřebujete informace o zákazníkovi, platforma skýtá aplikace s možnostmi doručit tyto informace na mobilní zařízení, opět formou webových XML služeb s bohatými komunikačními možnostmi.
- „Poskytuje tak zjednodušený vývoj, bohatou zkušenost uživatelů“:
  Pokročilé techniky, které pohotově zpřístupňují informace, jsou základem pro řešení, jenž poskytnou klientům plné uspokojení, nezávisle na tom, jak nebo z jakého zařízení k těmto informacím přistupují.
- Při používání technologií a možností zabudovaných do CLR, Visual Studia.NET, .NET SDK a .NET jazyků dojte také k významnému zlepšení zkušeností vývojářů. Více informací o způsobu jak vyvíjet a jak získat prospěch z těchto technologií najde v souvisejícím článku *„Proč Microsoft .NET?“*
- „Nebývalé příležitosti pro obchod“:
  Technologie a výhody pro vývojáře otevírají nové příležitosti pro řešení obchodních procesů, funkcionality a aplikací. Se servery jako je Microsoft BizTalk™ Server 2000 můžete nyní integrovat vaše obchodní procesy s procesy vašich partnerů a vylepšovat tak efektivnost, snižovat náklady a poskytovat skutečné výhody vašim zákazníkům.

Platforma .NET

Z technologického pohledu je Microsoft .NET platformou a zkušenostmi uživatelů postavených na základech této platformy. .NET platforma zahrnuje komponenty, které spojují dohromady .NET produkty a technologie sdílející společné základy postavené na otevřených Internetových standardech.

Platforma obsahuje:

- .NET Framework a nástroje Visual Studia.NET,
- .NET základní stavební služby,
- klientský .NET software pro chytrá zařízení,
- .NET Enterprise servery.

Analyzujme každou z těchto komponent:

.NET Framework a nástroje Visual Studia.NET

Jde o programový model, vývojové prostředí a nástroje pro vytváření nové generace distribuovaných aplikací. Vývojové nástroje jako je Visual Studio.NET maximálně zjednodušují vytváření webových XML služby a webové aplikací. .NET Framework a Visual Studio.NET doplněny o .NET Software Developers Kit (SDK) nabízí nejlepší, nejsnadnější, nejrychlejší a nejméně nákladný způsob, jak psát webové XML služby.

.NET základní stavební služby

Základní .NET stavební služby budou nabízet následující:

- **Identifikační službu.** Je založena na technologiích Microsoft Passport a Windows Autentikaci a poskytuje rozmanité úrovně autentikace počínaje hesly a konče smart kartami a biometrickými zařízeními. Vývojářům umožňuje produkci služeb, které zajišťují personifikaci a soukromí svým zákazníkům.
- **Služba zpráv a upozornění.** Slouží k integraci přímého zasílání zpráv, elektronické pošty, hlasové pošty a dalších forem upozorňování a zasílání zpráv do jednotné formy, dostupné na libovolném PC nebo chytrém zařízení. Je postavena na základech webové služby Hotmail, Exchange a MSN Instant Messenger.
- **Personalizační služba.** Poskytuje uživatelům možnost, jak aktivně řídit a vytvářet pravidla a nastavení, která implicitně a explicitně definují, jak mají být upozornění a zprávy zpracovány, jak se má zacházet s požadavky na sdílení vašich dat a jak mají být koordinována různá mobilní zařízení.
- **Kalendářové služby.** Kritickým aspektem v osobním managamentu je čas. Kdy je vhodné, aby vás někdo vyrušil, a kdy by jste měli zůstat nerušeni? Je to zejména důležité ve chvíli, kdy lidé začínají více používat zařízení a vzniká mnohem intenzivnější komunikace mezi uživatelem a službami. Microsoft .NET nabízí základy pro bezpečnou a soukromou integraci vaší práce, společenského a domácího kalendáře tak, aby byl dostupný na všech typech zařízení a samozřejmě, s vaším svolením, i jiným službám a jedincům. Tato služba je postavena nad službami Microsoft Outlook© messaging and collaboration klientů a Kalendáře služby Hotmail.
- **Adresářové a vyhledávací služby.** Microsoft .NET umožňuje vyhledávat služby a lidi, se kterými chceme komunikovat. Adresáře Microsoft .NET jsou mnohem více než vyhledávacím strojem nebo „zlatými stránkami“. Mohou programově spolupracovat se službami a zodpovědět schematizované dotazy o možnostech těchto služeb. Mohou být také interně využívány a přizpůsobovány jinými službami.
- **Dynamická dodávková služba.** Umožňuje Microsoftu a vývojářům nabídnout inkrementální úrovně funkcionality a spolehlivé automatické upgrady na vyžádání, bez nutnosti uživatelského zásahu při instalaci nebo konfiguraci.

.NET software pro zařízení a chytrá zařízení

.NET software pro chytrá zařízení dává k dispozici personálním počítačům, pracovním stanicím, chytrým telefonům, počítačům do ruky, Tablet PC, herním konzolám a dalším chytrým zařízením možnost fungovat ve světě .NET.

Chytrým zařízením se rozumí, když:

- Zná vás. Používá vaši .NET identitu a data vašeho profilu pro zjednodušení svého ovládání.
- Zná síť. Reaguje na omezení přenosového pásma, poskytuje podporu jak pro online, tak offline použití aplikací a zná dostupné služby.
- Zná informace. Může přistupovat, analyzovat a pracovat s daty kdykoli a kdekoli.
- Zná další zařízení. Vyhledává další PC, chytrá zařízení, servery, Internet a ví, jak poskytovat služby dalším zařízením. Chytré je také proto, že umí přistoupit k informacím na PC.

- Zná software a služby. Aplikace a data jsou prezentována optimálně pro konkrétní formát zařízení. Metody realizace vstupu a konektivity odpovídají aktuálnímu typu zařízení a poskytují skvělou interakci s uživatelem. Konzumuje webové služby s použitím XML, SOAP, UDDI a vývojáři jej mohou programovat a rozšiřovat.

Mezi software pro chytrá zařízení, na kterém Microsoft pracuje, patří Windows XP, Windows Me, Windows CE, Windows Embedded, .NET Framework a .NET Compact Framework.

Microsoft .NET Enterprise servery

Microsoft .NET Enterprise servery představují nejrychlejší a nejspolehlivější způsob, jak v podnicích integrovat, řídit a zpřístupnit aplikace do Webu. Microsoft dodává podnikové servery poskytující spolehlivost, výkonnost, růst a řiditelnost, tolik potřebné v průmyslových řešeních. Podpora v globální síti ve spolupráci s širokou množinou průmyslových partnerů poskytuje úplná internetová a intranetová řešení. Následující seznam skýtá přehled všech .NET Enterprise serverů. Více informací o těchto produktech lze získat na adrese: ***http://www.microsoft.com/servers***.

- Microsoft Application Center 2000. Jde o nástroj pro management aplikací, který umožňuje řídit aplikace na skupině serverů s jednoduchostí řízení aplikací na jednom počítači. Realizuje správu aplikací, možnosti škálování software a dostupnost u kritických aplikací, pomocí implementace takových koncepcí jako jsou webové farmy, škálování do šířky a clusterování. Nejde pouze o webové aplikace, ale také o aplikace na střední vrstvě a komponenty.
- Microsoft BizTalk Server 2000. Sjednocuje v jednom produktu technologie, které podporují Enterprise Application Integration (EAI) a Business-to-Business (B2B) integraci. Poskytuje také pokročilou technologii Orchestrace, která dává k dispozici obchodním analytikům, IT profesionálům a vývojářům možnosti vyvíjet dynamické obchodní procesy překračující hranice aplikací, platforem a společností. Pro zajištění interoperability používá otevřené architektury, umožňuje zasílání dat a zpráv mezi aplikacemi jak uvnitř, tak ven ze společností. Skýtá nástroje pro datové transformace, auditovaní, sledování a analýzu, společně se sofistikovanou bezpečností založenou na kryptografii s veřejným klíčem, ústředním požadavkem B2B řešení. Pro společnosti s existující EDI infrastrukturou BizTalk podporuje zpracování dokumentů ve formátu ANSI X12 a UN/EDIFACT EDI.
- Microsoft Commerce Server 2000. Realizuje aplikační rámec pro vývoj specifických řešení elektronického obchodu. Obsahuje pokročilé postupy pro realizaci mechanizmů zpětné vazby potřebné pro analytická zpracování. S tímto nástrojem lze vytvářet portály s optimální zákaznickou odezvou formou personifikovaného obsahu a podporovat online obchody vytvářející užší partnerské vztahy. Obsahuje také profilování a správu zákazníků, správu produktů a služeb, transakční zpracování a cílený marketing a inzerci, potřené pro efektivní řešení typu business-to-customer (B2C) a business-to-business (B2B). Je dodáván s hotovými aplikacemi pro B2B obchody a elektronickými obchody pro koncové zákazníky. Tyto aplikace mohou byt rozšiřovány a přizpůsobovány vlastními vývojáři, například přidáním různých softwarových řešení třetích stran, jakými mohou být validace kreditních karet, daňové systémy, dodavatelské služby nebo doprava.
- Microsoft Exchange Server 2000. Vedle realizace prvotřídního systému pro management zpráv, Exchange také realizuje spolupráci odborných pracovníků, integraci návrhu Webu a workflow aplikací, jednotnou infrastrukturu a uživatelský model pro práci se zprávami, dokumenty a aplikacemi. Poskytuje taktéž komunikační infrastrukturu pro přístup k informacím v libovolném čase a z libovolného místa pomocí nastupujících technologií, jakými jsou bezdrátová komunikace, sjednocené předávání zpráv, zařízení do ruky a telekonference.

Produkt je plně integrován se službami Active Directory Windows 2000, a podporuje clusterování a násobné databáze zpráv. Podporuje také distribuované služby, nativní Internet Mail Content a zajišťuje skupinové plánování času, diskusní skupiny, skupinové záložky, společně s vlastností indexování a vyhledávání. Je také charakteristickým nástrojem pro workflow management, webovým skladištěm dat jako instalovatelný souborový systémem, systémem pro jednotné a přímé předávání zpráv, službami chatu a v neposlední řadě datovým, audiovisuálním konferenčním systémem.

- Microsoft Host Integration Server 2000. Realizuje komunikační, aplikační a datovou integraci s mainframe systémy a systémy střední velikosti. Podporuje velké množství síťových protokolů a síťových služeb, uživatelský klient/server přístup a tisk vzdálených souborům a je charakteristický škálovatelností a výkonností založenou na interních možnostech Windows 2000 (jako je např. vyvažování zátěže – Load balancing). Umožní vám webový přístup k exitujícím mainframe aplikacím, přístup k DB2 databázím a souborovém systému na mainframech, AS/400 a Unix systémech. Také zvládá transakční integraci s prostředími CICS firmy IBM nebo IMS transakcemi a umožňuje tak kombinaci zpracování dat na mainframe systémech se zpracováním na platformě Windows.
- Microsoft Internet Security and Acceleration (ISA) Server 2000. Poskytuje bezpečnou a rychlou konektivitu do Internetu, integruje rozšiřitelný vícevrstvý podnikový firewall a vysoce výkonnou a škálovatelnou webovou cache. Realizuje ochranu sítě před neautorizovaným přístupem, zkoumá provoz a upozorňuje administrátory na případné útoky. Vyznačuje se bezpečností s několika úrovněmi monitorování provozu, integrovanými virtuálními privátními sítěmi (VPN), detekcí průniku, inteligentními aplikačními filtry, bezpečným publikováním serverů a extenzemi pro škálování do šířky pomocí clusterů a vyvažování zátěže. Jelikož žádný produkt nesplňuje všechny bezpečností a výkonnostní požadavky, je ISA server 2000 vybaven značnou možností rozšíření pomocí SDK (Software Development Kit) na úrovni firmy, nebo třetími stranami.
- Microsoft Mobile Information Server 2000. Jde o platformu rozšiřující záběr Microsoft .NET Enterprise aplikací, firemních dat a internetového obsahu do oblasti mobilních uživatelů. Spojuje navzájem bezdrátové aplikace a zařízení různých výrobců a mobilní operátory. Outlook® Mobile Access je portálem pro mobilní telefony poskytující prohlížení a zpracování upozornění z elektronické pošty, kalendáře, úkolů a kontaktů programu Outlook. Poskytuje také potřebné vývojové nástroje pro rozšíření existujících aplikací na mobilní zařízení.
- Microsoft SQL Server 2000. Jde o kompletní systém pro management relačních databází a systémovou analýzu, využitelný při realizaci elektronického obchodu a datových skladů. Jako data warehouse nástroj nabízí schopnosti integrace, konsolidace a shrnutí informací z různých datových zdrojů. Řídícím pracovníkům umožňuje sjednocovat data pro potřeby analytických nástrojů, které vyhodnocují vzory a trendy, předpovídají budoucí chování a vylepšují obchodní rozhodnutí.
  Pro podporu datové analýzy nabízí SQL server nástroje pro dolování dat, služby Online Analytical Processing (OLAP), bezpečnostní rozšíření a schopnosti přístupu k datovým krychlím z prostředí klientských prohlížečů (přes Internet nebo Intranet). Pro potřeby aplikací elektronického obchodu podporuje internetové standardy jako XML a nabízí programovou podporu interoperability s existujícími řešeními.

- Microsoft SharePoint Portal Server 2001. Je znám pod kódovým jménem „Tahoe“ a jde o komplexní a pružné portálové řešení usnadňující jednoduché vyhledávání, sdílení a zveřejňování informací. Tento nový server zahrnuje schopnosti managementu znalostí, bezproblémovou spolupráci s Microsoft Office a dalšími Microsoft produkty pro desktopové prostředí. Zmíněná integrace napomáhá realizovat robustní správu dokumentů, včetně vyhledávání a subskripcí. Taktéž podporuje online diskuse jako rozšíření managementu dokumentů a procesům spolupráce. SharePoint Portal server navíc nabízí podnikům, díky architektuře postavené na rozšiřitelné webové technologii, nejefektivnější způsob, jak vybudovat a rozšiřovat webové portály.

.NET User Experiences

.NET user experiences (prožitky s platformou .NET) jsou prostředky pro uživatelskou interakci s .NET aplikacemi. Jedna skupina těchto nástrojů navzájem propojuje webové služby a integruje jejich funkcionalitu, aby mohla poskytnou cílovým uživatelům, zákazníkům a vývojářům patřičné prožitky. Microsoft právě nyní transformuje čtyři populární produkty na .NET Experiences:

- Microsoft Office XP dělá první krok směrem k realizaci .NET Experience, zaměřené na lidi pracující s informacemi.
- MSN® ve spolupráci s lokálním klientem MSN Explorer je ve stádiu realizace zákaznicky orientované .NET Experience.
- bCentral™ jako malý obchodní portál realizuje potřebné webové XML služby pro malé firmy (jako je řízení zásob) a konzumuje důležité webové XML služby (jako je eBay).
- Vývojový systém Visual Studio® poskytuje vývojářům .NET Experience formou zpřístupnění informací z MSDN a dalších specifických firemních vývojářských směrnic přímo ve vývojovém nástroji.

Druhá skupina .NET User Experience leží v oblasti klientského programového vybavení (např. operační systémy a prohlížeče zákazníků). Tyto nástroje mají za úkol nabízet bohatší způsoby interakce napříč počítači a zařízeními, a zahrnují:

- vylepšené uživatelské rozhraní,
- rozpoznávání řeči,
- integrované komunikace,
- rozpoznávání písma.

.NET User Experiences má následující charakteristiky:

- Jsou poskytovány napříč mnoha zařízeními. Místo vytváření odlišných webových XML služeb a odlišných aplikací pro každé případné .NET zařízení, aplikace mohou přečíst charakteristiky těchto zařízení, nastavení způsobu přístupu koncového uživatele a použít odpovídající rozhraní.
- Využívají výhody přístupu k webovým XML službám ve chvíli, kdy jsou připojeny k síti a pro své uživatele sbírají dodatečné a relevantní informace potřebné k řešení každodenních problémů.
- Jsou orientovány na uživatele. .NET Experiences jsou zaměřeny na koncové uživatele tím, že využívají základní stavební služby pro jejich identifikaci, osobní nastavení, upozorňování a uživatelská data. Jelikož jsou uživatelská data spravována základními stavebními službami místo aplikací samotných, uživatelé mají kontrolu nad svými daty, mohou zajistit jejich správnost a mohou je sdílet mezi různými aplikacemi a službami.

POSÍLENÍ OBCHODU

Microsoft .NET vám dává k dispozici klíčové vlastnosti, pomocí nichž budete schopni dosáhnout významného obchodních prospěchu.

Tato část pojednává o některých obchodních výhodách .NET. Popisuje například, jak mohou obchodní analytici využít technologii .NET orchestrace pro počáteční modelování a následující automatizaci celého obchodního procesu, typicky obsahujícího interakci s aplikačními systémy jak uvnitř, tak vně společnosti, za účelem dosažení vyšší úrovně efektivnosti při správě obchodu.

Obchodní výhody

Vedle obchodního tlaku, kterému většina podniků v současnosti čelí (např. tlak na dobu uvedení výrobku na trh, akvizice a obchodní fůze, zvýšená závislost na externích dodavatelích, posuv k globálnímu obchodu 24x7), dochází k základním transformacím technologického prostředí, v němž obchod probíhá. Nepřetržité zvyšování výpočetního výkonu a konektivity, množící se zařízení a touha po standardizaci se spojují a vytvářejí obrovskou vlnu základních změn, která se teoreticky hrne přes každé odvětví a mění vnitřní struktury a předpoklady všech obchodů.

Enormní potenciál Internetu, doplněný o jeho schopnost průniku, otevírá mnoho nových příležitostí a nabízí až doposud netušené možnosti při zavádění novinek. Ke zcela novým inovacím dochází v hudebním průmyslu, kdy webové portály nabízejí služby pro stažení a sdílení hudby, v cestovním průmyslu, v makléřství, v oblasti realit a v moha dalších průmyslových sektorech.

Transformace přetváří některé základní předpoklady ve vašem obchodním modelu mnohem hlouběji, než pouhým přidáním webového rozhraní k vašim stávajícím a v podstatě nezměněným obchodním aplikacím. Společnosti, které chtějí přežít transformaci v digitální ekonomiku, potřebují připravit přehodnocení svého obchodu ve světle následujících třech nezbytných předpokladů:

- **Vytvoření propojené infrastruktury.** Infrastruktura neznamená pouhé čerpání maximální hodnoty z interních systémů logickým a jednotným použitím všech dostupných systémových dat, nýbrž také rozšíření vaší infrastruktury za zdi podniku, směrem k partnerům a dodavatelům. Jednotný tok dat napříč celým obchodním procesem (nezávisle na hranicích organizace) poskytuje významnou příležitost pro zjemnění a znovuvytvoření obchodních procesů s masivním zvýšením jejich efektivity. Vedle základních vylepšení existují příležitosti jak založit zcela nové obchody, o kterých nebylo možné dříve uvažovat, díky nesouladům v neintegrovatelné infrastruktuře.
- **Zplnomocnění vašich zaměstnanců.** Znamená to zpřístupnění všech dostupných dat vašim zaměstnancům, které jim umožní vytvářet lepší rozhodnutí a realizovat za vás mnohem efektivnější kroky. Znamená to také zpřístupnění dat v odpovídajících souvislostech, ve správném čase, správným zaměstnancům, a všude, kde jsou připojeni do vaší sítě. Více než prosté prohlížení a zpřístupnění informací, to znamená přesné určení relevantních informací pro každého zaměstnance, v každé fázi pracovního dne, nezávisle na tom, odkud tyto informace pochází (z jakého interního nebo z externího zdroje) a jejich shromáždění do místa, odkud zaměstnanec může provádět efektivní, správné a bezprostřední rozhodnutí.

- **Integrace do každodenního života vašich zákazníků.** Tento předpoklad má jeden z největších důsledků pro obchod. Prověřte znovu, napříč celým vaším odvětvím, kdo jsou vaši zákazníci a jaké základní hodnoty jim poskytujete. Nyní si představte ideální zkušenost každého koncového uživatele, kterou získá, když mu dodáte přesně ty hodnoty, jenž potřebuje a to tím nejvhodnějším způsobem, plně integrovaným do jeho každodenního života a prostředí. Představte si cílenou nabídku služeb těmto klientům, v pravou chvíli a přizpůsobenou přesně na požadovanou míru. To je radikálně odlišný způsob vedení obchodu, než jaký provozujeme v současnosti.

Microsoft vytváří .NET platformu pro usnadnění transformace vašeho obchodu, jako reakci na uvedené požadavky. Je založena na modelu webových služeb, které jsou přirozeným důsledkem nárůstu výpočetního výkonu, konektivity a posunu směrem ke standardům, které charakterizují celý technologický průmysl.

Microsoft .NET je navržen tak, aby vám pomohl vytvořit dostatečně svižný obchod, který přežije transformaci vašeho průmyslu. Poskytuje vám nástroje pro formulaci rychlé odpovědi na změny v obchodu a v průmyslu kolem vás a současnému využití jejich výhod a možností k vlastnímu růstu. Podpora pružné integrace uvnitř i napříč společnostmi je základní součástí této platformy. Tím, že vytváří nástroje umožňující zaměstnancům pořizovat a manipulovat s daty, Microsoft stojí v čele průmyslu. Pokrok v mnohých z těchto nástrojů vám významně napomůže při řešení úkolu zplnomocnění vašich zaměstnanců. .NET technologie vám také umožní využít znalosti vašich zákazníků, nabízet jim nejvhodnější produkty a služby v patřičnou chvíli a v nejvhodnějších souvislostech.

Každý obchod se bude odehrávat v této digitální ekonomice a otázku, kterou by jste si měli klást zní: „Jak by měl můj obchod vzájemně reagovat s ostatními elektronickými obchody?“. Microsoft vlastní technologie, výpočetní model a platformu, která vám pomůže lukrativně vybudovat celé řešení.

Hodnota pro obchod

Internet dává společnostem k dispozici rostoucí platformu pro realizaci portálů, které inzerují, propagují a prodávají zboží a služby. Microsoft .NET rozšiřuje tuto platformu a umožňuje portálům vzájemné propojení pomocí webových XML služeb. Umožní vám dosáhnout vyššího stupně synergie s vašimi klienty a partnery a integrovat vaše dodavatelské řetězce.

Webové služby

Představme si scénář, v němž klient hledá produkt, porovnává případné výrobce použitím webových portálů, nakupuje produkt od zvoleného výrobce a pak čeká na jeho dodávku. V příkladu mohl zvolený výrobce poskytnout zákazníkovi detailní informace o doručovací společnosti, která zboží dodá. V takové chvíli je zákazník schopen přistoupit na webové stránky doručovací společnosti a objednávku sledovat.

Microsoft .NET umožňuje posunout tento proces o jeden krok dále. Pomocí webových XML služeb je výrobce produktu schopen přímo přistoupit k dodávkové společnosti a její aplikaci pro sledování zakázek a získat tak informace jako jsou aktuální stav dodávky, aktuální pozice zboží a plánované datum a čas dodávky. Integrace mezi výrobcem a dodavatelskou společností poskytuje výrobci možnost doplnit informace na svém Webu a směrovat je ke svým zákazníkům. To je nesmírná výhoda pro zákazníka, který může nyní sledovat dodávku přímo na webovém portálu dodavatele (a ne na několika odlišných webových portálech). Zákazník nyní získá mnohem lepší a lépe fungující zkušenost z nakupování.

Microsoft .NET vám nejen zprostředkovává publikování informací o vašem zboží a službách na webovém portálu, ale také zveřejňuje webové služby, které mohou být použity na webových portálech jiných společností (a naopak). Například, pokud vaše společnost prodává hudební CD, můžete zveřejnit webovou službu, přes níž si vaše vlastní nebo partnerské obchody mohou žádat informace o zásobách online formou přes Internet. Pokud zákazník ve vašem partnerském obchodu později vytvoří objednávku, obchod použije jinou z vašich webových služeb pro automatické zadání objednávky. Na základě přijaté objednávky budete schopni automaticky zavolat webovou službu vaší dodávkové společnosti a dokončit obchodní proces.

Webové služby vám napomohou v implementaci průmyslového JIT (Just-in-time) modelu zásobování dodáním větší integrační pružnosti a rychlosti prodejním řetězcům. Efektivní JIT management může napomoci snížit výdaje, redukovat výrobní dobu vašich zakázek a udržet optimální úroveň zásob. Doplněno o automatický proces dodávky a zásobování dojde k celkovému zlepšení finančního řízení a zákaznických služeb.

Webové služby také poskytují příležitosti při budování a podpoře nových partnerských vztahů, vytváření nových a vylepšených obchodních modelů a povýšení vaší konkurenceschopnosti na trhu. Microsoft .NET se zaměřuje na realizovatelnost všech těchto cílů.

Webové portály
Microsoft .NET nabízí vývojářům a návrhářům tradičních internetových webových portálů pokročilé nástroje, které významně napomohou a urychlí jejich generování. Je zde integrována podpora pro různé typy prohlížečů, tradiční a velice obtížně řešitelný, časově náročný problém. Zejména je důležité, že dochází k vytváření aplikací a webových portálů v mnohem kratším čase, rychleji než u konkurence a aplikace jsou směřovány na širší oblast trhu. Časová úspora vám umožní vylepšit služby pro nové zákazníky.

Komunikace
Microsoft .NET poskytuje ještě další obchodní výhodu. Komunikace představují jeden z nejvyšších nákladů spojených s distribuovanými společnostmi. .NET interně používá otevřené průmyslové standardy (jako HTTP, XML a SOAP), které vám umožňují připojovat sesterské firmy a pobočky ke společným službám prostřednictvím Internetu jako transportu, při redukci komunikačních nákladů a poskytnutí rychlého připojení do online služeb novým prodejním místům. Vaše obchodní plány tím získají velkou pružnost. Internetové standardy jsou optimalizovány pro pomalé a nespolehlivé linky. Možnosti nasazení těchto technologií a standardů pro připojení vašich poboček je obrovskou výhodou.

Microsoft .NET poskytuje první a nejkompletnější platformu, která vám umožňuje vytvářet nové interní aplikace, integrovat vaše stávající nástroje a aplikace, zakládat webové portály, stavět webové služby a využívat webové služby jiných společností pro budoucí růst vlastního obchodu.

Přínos pro zákazníky

Microsoft .NET je důležitý i pro koncové zákazníky, protože zjednodušuje používání aplikací a nabízí mnohem více funkcionality. Konkrétně oprošťuje uživatele od umělých hardwarových omezení. Uživatelská data žijí v Internetu nebo ve firemní síti, nikoli pouze na uživatelském osobním počítači nebo menším zařízení. Lze k nim přistupovat z jakéhokoli stolního počítače, přenosného počítače, mobilního telefonu, Pocket PC a jsou plně integrována ve všech aplikacích.

Microsoft .NET umožňuje uživatelům rychle sestavit spojení a dokončit transakce, což až do posud vyžadovalo hodiny a frustrující opakované vkládání dat z jedné aplikace do druhé. Tím, že lze kombinovat několik bezpečných datových zdrojů do jednoho uživatelského rozhraní (nebo dokonce programovatelného rozhodovacího nástroje), .NET platforma oprošťuje uživatele od omezení způsobených postupy, jakými jsou v současnosti data v Internetu publikována.

Úplná volnost v přístupu k datům uživatelů a zaměstnanců z libovolného místa a v libovolném čase vytváří úžasné prostředí, které inspiruje nové obchodní příležitosti.

Automatizace obchodních procesů

BizTalk Server 2000 spolu s technologií orchestrace a nástrojem Application Designer poskytují obchodním analytikům značné výhody a umožňují jim graficky definovat a modelovat obchodní procesy. Nástroj Application Designer umožňuje dynamické modelování procesů, které je na hony vzdálené vytváření statických obchodních modelů. Tyto modely definují datové toky mezi interními a externími aplikacemi a pomocí obchodní logiky přesně definují, jak mají být tyto aplikace navzájem propojeny. Navíc může orchestrační jádro spustit model, implementovat definované obchodní procesy a aplikovat rozmanité vývojářem definované a implementované technologie, mezi jinými také webové služby.

Abychom ilustrovali schopnosti BizTalk orchestrace a demonstrovali jeho přitažlivost pro obchodní analytiky, uvažujme následující příklad, který byl prezentován na PDC (Professional Developers Conference) v roce 2001.

Ve scénáři volá Standa do ordinace svého lékaře a hovoří s Janou, jeho sestrou. Žádá ji o objednání k návštěvě, neboť si zlomil nohu. Pomocí dvou kliknutí myši je Jana schopna objednat Standu ve vhodnou dobu. Zdá se to tak jednoduché, ale tento jednoduchý příklad představuje o čem vlastně webové služby jsou.

V pozadí těchto dvou kliknutí myši existuje samozřejmě velké množství webových služeb a aplikací, které spolu navzájem reagují.

Při trochu podrobnějším rozboru uvedeného scénáře vyplyne následující sekvence akcí provedených jednotlivými lidmi, webovými službami a aplikacemi zahrnutými ve scénáři:

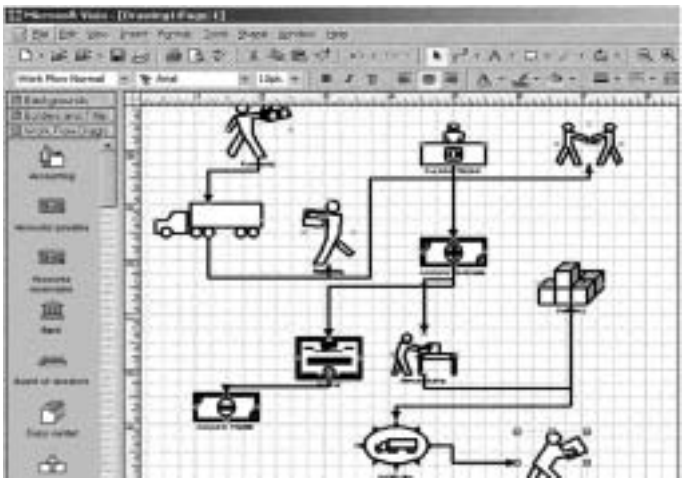
- Jana přistupuje k osobním informacím Standy a získává seznam pojišťovacích společností, u kterých je Standa veden.
- Zasílá tento seznam, společně s jeho aktuální pozicí do kartotéky pacientů kliniky. Kartotéka pacientů je webová služba, která stojí v samém srdci tohoto scénáře.
- Kartotéka pacientů kliniky vrací seznam dostupných termínů zpět do Janiny aplikace a Jana je tlumočí Standovi.
- Standa si následovně jeden vybere a Janina aplikace zasílá požadavek na schůzku zpět webové službě.

Služba začne vykonávat sérii dalších kroků:

- Upozorní patřičného lékaře o potřebě operovat Standu.
- Pošle zpět potvrzení do Janiny aplikace a ta oznámí tuto informaci Standovi.
- Zašle upozornění Standově pojišťovací společnosti a oznámí jí přijetí pacienta.
- Nakonec zašle informaci o nemocenské Standovu zaměstnavateli.

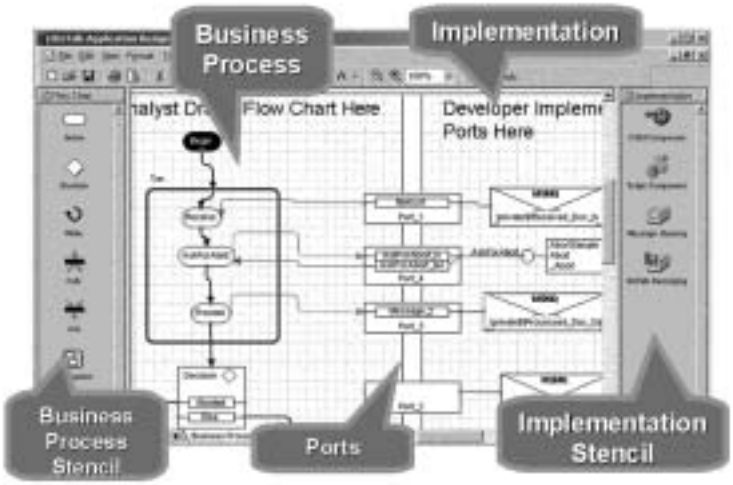
Tento scénář demonstruje velké množství webových služeb, které spolu navzájem reagují velmi složitým způsobem. Komunikují spolu pomocí webových XML služeb a spolupracují s dalšími aplikacemi. Vybudovat tradičním způsobem aplikaci jako je kartotéka pacientů kliniky je velice složitý úkol. Zabrala by velké množství kódů společně s vysokým počtem velmi zručných a zkušených programátorů, kteří by museli pracovat týdny nebo i měsíce a postupně se přibližovat ke kýženému výsledku.

Při použití nástroje BizTalk Application Designer je však vytvoření takového obchodního procesu drasticky zjednodušeno. Application Designer je postaven nad produktem Microsoft Visio 2000. Jde o velkou výhodu pro mnoho obchodních analytiků, kteří tento nástroj již znají a používají jej při každodenním kreslení obchodních diagramů. Obrázek 7 ukazuje typický diagram obchodního procesu, který analytici v současné době vytvářejí.



Obrázek 7 Obchodní proces popsaný ve Visiu

Application Designer posouvá tento přístup o jeden krok dále a umožňuje obchodním analytikům vytvářet spustitelné aplikace přímo z jejich procesních modelů při použití známých metodologií a nástrojů. Application Designer rozděluje svou návrhovou plochu na dvě části, jak vidíme na obrázku 8.



Obrázek 8 Vývojové prostředí BizTalk

Na levé straně obchodní analytik vytváří definici obchodního procesu pomocí sekvence elementů představujících jednotlivé akce, které definují požadovanou sekvenci kroků a vedou k naplnění obchodního procesu. Akce jsou vytvářeny přetažením Visio elementů ze šablony obchodních procesů umístěné nalevo. Následovně propojí elementy dohromady pomocí spojovacích šipek definujících korektní sled událostí.

Jakmile je model definován, obchodní analytik jej předá vývojáři. Vývojář použije pravou část pro specifikaci implementačních elementů (z implementační šablony), která definuje, jak má být každý akční element (definován obchodním procesním modelem) implementován za běhu aplikace. Vývojář si může vybrat z široké nabídky technologií včetně COM komponent, Windows skriptovacích komponent, Windows front zpráv a BizTalk Messaging. (BizTalk Messaging, realizovaný BizTalk Serverem 2000, poskytuje různé transformace zpráv, mapování dat, auditování, transportní a bezpečnostní vlastnosti potřebné pro budování business-to-business (B2B) a Enterprise Application Integration (EIA) řešení). Vývojáři budou také často volit nasazení webových služeb pro implementaci jednotlivých akcí definovaných v obchodním modelu.

Jakmile je zřejmé, jak budou jednotlivé akce v obchodním procesu implementovány, vývojář může propojit implementační elementy vpravo s akčními bloky na levé straně. V této chvíli také může definovat, jak budou data (obvykle XML zprávy) proplouvat skrz procesní model. Závěrem, ve chvíli, kdy všechna propojení byla provedena, je model přeložen a vytvořen orchestrační plán. Ve skutečnosti jde o XML reprezentaci obchodního procesu (nazvaného XLANG plán), který obsahuje všechny požadované akce, přesnou sekvenci událostí a obsahuje přesnou specifikaci mapování modelu na konkrétní implementované technologie. Jako příklad si uveďme element spojený s frontou zpráv ve Windows nakonfigurovanou pro příjem. Časový plán pak bude monitorovat frontu na příchod specifikované zprávy. XLANG časový plán je následovně zpracován BizTalk jádrem pro orchestraci, které spustí aplikace a přivede obchodní model k životu.

Tento radikálně odlišný přístup k návrhu aplikací ilustruje klíčovou myšlenku. Vývoj v současnosti obvykle zahrnuje integraci existujících aplikací. Ve většině případů nechcete vyvíjet ohromené množství nového kódu. Místo toho aplikační vývoj zahrnuje integraci existujících aplikací jako je elektronická pošta, systémy pro řízení vztahů se zákazníky (CRM), aplikace Enterprise Resource Planing (ERP), mainframe aplikace atd. Velmi často také zahrnuje aplikace jako message queuing, komponenty nebo webové služby, které jste mohli vyvinout ve Visual Studiu.

Uvažte, jak by mohl být dříve zmíněný obchodní proces vyvinut s pomocí BizTalk orchestrace. Pro vytvoření obchodního modelu na levé straně návrhové plochy, propojte navzájem akční elementy reprezentující obchodní proces. V případě zmíněného scénáře, první akční element by byl označen jako „Přijmout požadavek k návštěvě“ (jde o požadavek, který přijme registr pacientů kliniky od Janiny aplikace v ordinaci lékaře). Druhý akční element by byl označen jako „Registrované objednávky“. Akční elementy by byly potřebné i pro každý další jednoznačný krok v obchodním procesu. Následovně musí být všechny propojeny v odpovídající sekvenci pomocí spojovacích šipek. Typický obchodní proces se bude skládat z mnoha akčních elementů.

Jakmile je postaven model, vývojář následovně vytvoří model na pravé straně pomocí specifikace odpovídajících implementačních technologií, požadovaných pro realizaci každé akce. První akce „Přijmout požadavek k návštěvě“, například, může být implementována jako webová služba vyvinutá pomocí Visual Studia.NET. Vedle integrace s webovými službami vám orchestrace poskytuje integraci s existujícími aplikaci jak firemními, tak aplikacemi uvnitř organizací obchodních partnerů. Např., element pro implementaci BizTalk messaging lze použít pro odesílání XML zpráv (nebo EDI zpráv) partnerským společnostem. Integrace s domácími systémy, jako jsou mainframe systémy, může být také dosaženo pomocí technologie message queuing. Vývojář má velkou volnost ve výběru technologie.

Po té, co byly přiřazeny implementační elementy ke všem akčním elementům obchodního procesu, lze model přeložit do orchestračního plánu. Orchestrační jádro ve výsledku spustí plán a implementuje kompletní obchodní model.

Reakce na změny

Dynamický a neutichající rozvoj vlastností modelovaných a implementovaných obchodních procesů je jednou z největších výzev, které dnes profesionálové čelí v oblasti IT. Společnosti musí soupeřit ve svých schopnostech přizpůsobit se změnám v obchodních procesech. S tradičními technologiemi a postupy je toho velice obtížné dosáhnout. Když se obchodní model změní, obchodní analytik v tradičním pojetí změní statický diagram modelu procesu a pak projednává detaily změny s vývojářem. Vyžaduje to většinou několik kol tohoto snadno omylného procesu, před tím, než je nová aplikace vyvinuta.

Přístup pomocí BizTalk orchestrace ulehčuje zapracování změn do obchodního procesu. Např. v dříve popsaném obchodním procesu, jsou potvrzující zprávy postupně zasílány lékaři a pak pojišťovacímu agentovi. Místo toho můžete požadovat paralelní provedení těchto činností. K dosažení změny stačí použít akčního element „větvení“, který umožňuje paralelní spuštění dvou nebo více akcí najednou. (Tvar „spoje“ je následovně použit pro synchronizaci paralelních větví běhu.)

Uvažujme nyní jiný typ změny, který může nastat. Když společnost vymění existující balík pro vedení účtů na mainframe systému za novou ERP aplikaci, implementační blok a port, který modeluje konektivitu do mainframe systému (pravděpodobně přes Message Queue element), lze odstranit a nahradit novým elementem (pravděpodobně COM komponentou poskytující konektivitu do ERP systému), který se zapojí do modelu. Designer a technologie orchestrace umožňují velmi snadné a rychlé provedení obou typů změn.

Uvedený příklad se dotknul některých vlastností podporovaných technologií orchestrace. Podrobnější informace o orchestraci a dalších vlastnostech jako jsou transakce umístěné do orchestračního postupu naleznete na stránkách ***www.microsoft.com/biztalk***.

POSÍLENÍ TECHNOLOGIE

Microsoft .NET podporuje vysokou úroveň integrace s existujícími infrastrukturami a aplikacemi. Zavádí nové technologie a rozvíjí existující, čímž vám umožňuje využívat exitující zkušenosti zaměstnanců. Využívá také stávající hardware a podporuje škálování do šířky a do výšky na základě vašich požadavků. Vývojáři mohou již dnes začít stavět webové aplikace pomocí SOAP toolkitu a Visual Studia 6.0 nebo ještě lépe pomocí právě uvedeného Visual Studia.NET.

Rozvoj Windows DNA 2000

Infrastruktura a .NET platforma je pokračováním dnešní Windows Distributed iNternet Architecture (Windows DNA 2000). Vaše existující investice, včetně znalosti vývojářů a softwarové základny jsou, ve svém důsledku, doplněny o nové schopnosti jako webové služby. Microsoft uvedl Windows DNA jako realizaci škálovatelné architektury pro řešení podnikového rozsahu a realizaci distribuovaných aplikací. Implementací n-vrstvého výpočetního modelu předepsaného modelem Windows DNA 2000, byly vývojáři schopni používat webové technologie ke snížení nákladů na nasazení a dosáhli úrovně škálování, která nebyla možná s předchozími typy architektur. Architekturami, které měly více monolitický a desktopově orientovaný přístup. .NET nemění logiku nebo funkcionalitu vaší aplikace samotné, nýbrž poskytuje aplikacím nové a pružnější způsoby zveřejnění jejich funkcionality ve formě webových služeb. Současné Windows DNA 2000 aplikace mohou být rozšířeny o funkcionalitu webových služeb, které mohou být následovně integrovány a orchestrovány s jinými webovými službami za použití platformy .NET.

Pokud vaše existující aplikace dodržují Windows DNA 2000 model, Microsoft .NET v mnoha ohledech usnadní jejich vývoj a také vám umožní spouštět nové aplikace vytvořené ve Visual Studiu.NET spolu s existujícími aplikacemi napsanými ve Visual Studiu 6.0. .NET dává k dispozici sofistikované služby pro spolupráci s existujícími aplikacemi a komponentami, které umožní jejich snadné nasazení. Pokud vaše společnost nenasadila architekturu Windows DNA 2000, Microsoft .NET poskytuje výkonnou infrastrukturu, která umožní vývojářům realizovat nové aplikace, které se dobře integrují s existující kódovou základnou.

Vývoj webových aplikací

Pro vývoj webových aplikací je k dispozici ASP.NET (pokročilejší verze současných Active Server Pages), která vám umožňuje realizaci tradičních aplikacích v prostředí prohlížeče, ale i webových služeb. ASP.NET se vyznačuje technologií umožňující jednoduchý vývoj webových stránek založených na serverových ovládacích prvcích, které automaticky generují odpovídající HTML, v závislosti na schopnostech klientského prohlížeče. Doposud se musela tato uzpůsobení provádět manuálně. ASP.NET šetří úsilí vývojářů a zajišťuje, že neztratíte své potenciální klienty.

ASP.NET obsahuje mnoho dalších pokročilých vlastností včetně vylepšené škálovatelnosti a výkonnosti, jednoduššího nasazení a mimořádných možností při řízení stavu připojení, uzpůsobených pro konfigurace s webovými formami.

Tím, že používáte.NET model webových služeb pro vývoj svých aplikací, vylepšujete způsob realizace podnikových aplikací. Spojení interních a externích služeb umožňuje zjednodušené vytváření aplikací, které integrují firemní data se souvisejícími daty prodejců a výrobců a mají za důsledek bezprecedentní šířku funkcionality jak pro koncové uživatele, tak zákazníky.

Nasazení aplikací

Termínem „xcopy nasazení“ označujeme jednoduchý způsob, jakým lze .NET aplikace nasadit. Pří instalaci aplikace jednoduše nakopírujte odpovídající soubory do zvoleného aplikačního adresáře. Nejsou nutné žádné konfigurační nástroje a registrace. Konfigurační nastavení jsou přenášena společně s aplikací, formou konfiguračních souborů založených na XML. Jednodušší nasazení a management nových verzí programového vybavení umožňuje IT odborníkům rychleji reagovat na měnící se obchodní požadavky.

Nový model nasazení také podporuje koexistenci různých verzí aplikace nebo komponent na stejném počítači a ve stejném čase. Například, nová verze webové aplikace může být nasazena vedle již existující. ASP.NET bude směrovat nové dotazy uživatelů na novou verzi, zatímco staří uživatelé budou obsluhováni starší verzí aplikace. Jakmile se dokončí obsluha existujících dotazů, stará verze aplikace je ukončena.

Nový model nasazení minimalizuje problémy spojené s uvolňováním nový verzí software díky zjednodušenému procesu instalace a ochrany před problémy spojenými s různými verzemi. To vše snižuje čas potřebný pro uvedení aplikace na trh.

Platforma pro Enterprise management

Primárním předpokladem Microsoft .NET je integrace aplikací. Management integrace je ve svém důsledku kritickým elementem úspěchu. Zákazníci mohou lépe sledovat a řídit své výpočetní systémy díky základnímu managementu distribuovaných výpočtů a .NET prostředí poskytovaných firmou Microsoft, která se stávají integrální součástí všech prostředí operačních systémů Windows. Windows Management platforma se skládá z:

- **Microsoft Operation Manager.** Zaměřuje se na management událostí a výkonu Windows 2000 serveru a .NET Enterprise serverů. Pro realizace operačního managementu Microsoft licencoval produkt Operation Manager firmy NetIQ, který je sám postaven nad management službami dostupnými ve Windows 2000. Microsoft produkt Operation Manager dále rozšířil a NetIQ nabízí další řešení pro nové Microsoft management produkty, včetně podpory jiných operačních systémů a aplikací jiných výrobců.
- **.NET Management Services.** Jde o rozšířenou množinu infrastrukturních služeb obsahujících management portál s podporou XML a SOAP, vytvořený pro realizaci nových řešení, která se integrují s .NET platformou.
- **Microsoft Management Alliance.** Jde o nový program napomáhající nezávislým výrobcům při tvorbě management produktů pro Windows platformu. Více informací o Microsoft Management Alliance najdete na ***http://www.microsoft.com/windows2000/management/articles/mmafaq.asp***

Znalostní báze a podpora ze strany Microsoft

Microsoft celosvětově poskytuje podporu nejen svých produktů, ale také svých strategií, řešení a průmyslových partnerů. Microsoft můžete kontaktovat a požádat o konzultační služby na různých úrovních během vývojové cyklu vašeho projektu. Setkat se lze s:

- Microsoft Account Manager, který rozumí obchodní strategii Microsoftu a může vás odkázat na další zákazníky Microsoftu za účelem získání referencí,
- systémovými inženýry (z firmy samotné nebo od Microsoft partnerů) nebo konzultanty (z Microsoft Consulting Services (MCS), nebo Certifikovaných partnerů), kteří vám poskytnou školení technologických strategií Microsoftu a informace o tom, jaké výhody může vaše firma získat při jejich nasazeních,
- partnery z průmyslu, kteří jsou informováni o hardwarových a softwarových řešeních založených na Microsoft technologiích aktuálně dostupných na trhu a mohou vám doporučit, jak využít jejich výhod.
- MCS nebo Microsoft partnery za účelem vytvoření návrhu a postavení strategie s Microsoft řešeními,
- různými skupinami zajišťujícími podporu a s partnery, řešícími specifické otázky podpory. Například, MCS, PSS (Product Support Services) a MSDN (specifická podpora produktů pro vývojové pracovníky).

Seznam s podrobnějšími informacemi o podpoře je dostupný z ***http://support.microsoft.com/directory***, nebo kontaktujete nejbližší Microsoft pobočku na ***http://www.microsoft.com/worldwide***.

Případně se podívejte na Microsoft Consulting Services v oblasti podnikových řešení na ***http://www.microsoft.com/trainingandservices/default.asp***.

O možné spolupráci s firmou Microsoft se dozvíte na adrese ***http://www.microsoft.com/business/microsoft/default.asp***.

VÝHODY PRO SPOLEČNOSTI

Integrace obchodu

Integrováním různých částí vaší infrastruktury (jak interní, tak externí) do struktury informací, které přesně odráží váš obchod a procesy, můžete si dovolit:

- Upevnit si svou základní pozici:
  - Orchestrace obchodních procesů dává jednotný pohled na vaše obchodní procesy napříč všemi platformami a zjednodušuje záměnu těchto obchodních procesů za účelem zvýšení efektivnosti.
  - Využít výhodu existujících investic do lidí a jejich schopností, místo hledání nových úzce specializovaných zaměstnanců s vysokými finančními nároky.
  - Snížit náklady a fůze a akvizice, integrovat systémy místo jejich duplikace nebo opakovaného vytváření.
  - Poskytnout jednotný pohled na zákazníky, který vám umožní nabízet kvalitní služby a mnohem lépe reagovat na požadavky zákazníkům.
- Zvýšit obchodní agilitu:
  - Na základě potřeby a pro uspokojení měnících se požadavků vašeho obchodu rozšiřovat systémy využitím velkého množství levného hardware a technologií škálování do šířky.
  - Technologie dynamické integrace vám umožňují pružně a rychle připojovat a pracovat s partnery a systémy na základě obchodní potřeby.
  - Použijte webové XML služby pro hlubší integraci s partnery, snížení nesouladů mezi obchodními procesy, zvýšení reakce schopnosti a výnosů.
- Povýšit technologii na konkurenční výhodu:
  - Využijte své schopnosti integrovat veškeré obchodní procesy a vytvářet nové hodnotné obchodní scénáře jako konkurenční výhodu, která není bez integrace možná.
- Integrujte rychle, jednoduše a levně:
  - Povyšte hodnotu svých existujících systémů jejich plnou integraci do vašich elektronických obchodních procesů a propojte je s vašimi novými aplikacemi a systémy.
  - Vnitřní podpora pro integraci ji činí rychlejší, levnější a jednodušší, než tradiční integrační úsilí nebo úplná záměna.
- Používejte nejlepší aplikace:
  - Využívejte nejlepší aplikace daného druhu pro dosažení maximální konkurenční výhody, kterou vám může aplikační infrastruktura poskytnout.
  - Rozšiřujte nebo nahrazujte systémy tak, jak vám obchod diktuje.
  - Vyberte si raději nejlepší z dostupných řešení než pouze průměrná řešení od jednoho výrobce. Nezdržujte se čekáním na vybraného výrobce než vám nabídne upgrade aplikací, které potřebujete. Vyberte si výrobce, který vám nabídne nejlepší řešení pro realizaci potřeb vašich firemních záměrů.
- Vyberte si nejlepší obchodní partnery:
  - Vyberte si nejlepší dostupné partnery, nezávisle na velikosti nebo obchodních aplikacích, které vaši partneři používají.
  - Pro své zákazníky buďte odlišní tím, že nabízíte kvalitnější služby formou integrace s partnery.
  - Vytvořte B2B infrastrukturu s kýmkoli a kdekoli to potřebujete.

- Nejlepší obchodní integrace:
  - Orchestrace obchodních procesů serveru BizTalk 2000 nabízí nejrychlejší způsob, jak integrovat a instrumentovat obchodní procesy.
  - Využití existujících schopností eliminuje dobu, kterou vývojáři potřebují ke studiu nových technologií a implementaci řešení.
  - Integrace aplikací místo jejich odstavení a záměny vám umožní věnovat bezprostřední pozornost rozšíření skutečných obchodních hodnot a významně posunout úměru čas-hodnota.

Posílení zaměstnanců

Posilujte produktivitu zaměstnanců prostřednictvím možného připojení kdykoli, kdekoli a z libovolného zařízení.

- Zprostředkujte lepší rozhodnutí:
  - Zaměstnanci mohou dělat rychlejší a lepší rozhodnutí, když jsou k problému mnohem blíže a pokud mají přístup ke všem relevantním informacím ze všech důležitých zdrojů.
- Využívejte chytrá zařízení:
  - Umožněte přístup a poskytněte možnost pracovat s informacemi na libovolném chytrém zařízení způsobem, který je pro dané zařízení nejvýhodnější.
  - Udělejte z mobility pracovníků konkurenční výhodu tím, že umožníte zaměstnancům vykonávat práci efektivně, z libovolného místa kam si mohou vzít přenosné zařízení nebo mobilní telefon.
- Podporujte samostatnost zaměstnanců:
  - Dejte místo kreativitě zaměstnanců pracujících pro vaši organizaci.
  - Vylepšete obchodní výkonnost tím, že dáte každému zaměstnanci příležitost pracovat na hlavních cílech.
  - Umožněte zaměstnancům rychle a přesně reagovat na příležitosti trhu pomocí nástrojů, která již důvěrně znají.

- Týmová práce a spolupráce
  - Umožněte spolupráci ve virtuálních týmech společně s partnery tak, že zaměstnanci kolektivně pracují na dosažení společného cíle.
  - Zaměstnanci mohou vzájemně komunikovat nezávisle na zařízení, které používají a místě, kde se nacházejí a to jak ve formě přímé komunikace, tak při sdílení dat a aplikací.
  - Workflow management se stává snadnějším a mnohem efektivnějším uvnitř i vně organizací, když data bezproblémově plynou od jednoho člověka nebo skupiny lidí k ostatním.
  - Sdílené informace hladce tečou z prostředí jednoho zaměstnance k jinému a umožňují průběžné zpracování informací a operace v rámci působnosti každého zaměstnance.
  - Jakmile jsou potřebná data bezproblémově přenášena a používána kýmkoli, kdo je potřebuje, dochází k lepší komunikace mezi zaměstnanci.

Orientace na zákazníky

Najděte nové zákazníky a spojte obchod se životem všech vašich zákazníků tak, že jim poskytnete odpovídající přístup k obchodním informacím způsobem a v době, který jim vyhovuje.

- Dosažení výnosu:
  - Lepší výnosy získáte nalezením nových zákazníků a zvýšením prodeje jak novým, tak existujícím zákazníkům formou přesnějšího směrování nabídky.
- Nalezněte a dostaňte se k zákazníkům. Umožněte, aby zákazníci nalezli vás.
  - Úsporně hledejte nové zákazníky pomocí programového zveřejnění vašich nabídek miliónům potenciálních klientů, jejichž preference a prostředí vám umožňuje přesně zacílit nabídku na ty zákazníky, kteří ji v danou chvíli očekávají.
  - Postupného navýšení výnosů dosáhnete spojením se zákazníky, kteří potřebují vaše produkty a služby, nejlépe v pravou chvíli a přes nejvýhodnější médium.
- Porozumějte a potěšte své zákazníky:
  - Poznejte své zákazníky a vytvořte si na ně a jejich potřeby mnohem komplexnější a integrovaný pohled.
  - Reagujte na požadavky a potřeby každého zákazníka přesně tím způsobem, který vyžaduje a který očekává.
  - Poskytněte zákazníkům plné informační služby, které využívají vše, co o nich a vašich vzájemných vztazích víte.
- Chápejte své zákazníky:
  - Prodávejte mnohem rychleji pomocí lepší orientace na klientské potřeby a požadavky.
  - Prodávejte mnohem rychleji pomocí styku se zákazníkem ve chvíli, kdy potřebují vaše produkty a služby.

ZÁVĚRY

Díky Microsoft .NET se mohou společnosti spolehnout na Internet jako na obchodní platformu. Microsoft .NET oslovuje většinu nedostatků současné výpočetní techniky a realizuje vizi podporující přístup ke všem uživatelským datům a aplikacím z libovolného místa a libovolného zařízení. Uživatelé jsou schopni pracovat se svými daty formou rukopisu, hlasu a obrazových technologií. Jejich data bezpečně žijí v Internetu nebo v Intranetu tak, aby k nim mohli přistupovat z domácích a firemních počítačů, z mobilních telefonů a pagerů, ze svých PDA a Pocket PC. Aplikace jsou schopny přizpůsobovat svou funkcionalitu omezením a možnostem zařízení, s nimiž uživatelé pracují. Aplikace jsou schopny za uživatele pracovat pomocí předdefinované množiny nastavení a instrukcí. Podniky tak mohou těžit z významně zvýšené efektivnosti a produktivity, protože .NET sjednocuje zaměstnance, zákazníky, informace a obchodní aplikace do logického a interaktivního celku.

Microsoft vize nové generace distribuovaných výpočtů na Internetu je postavena na poskytování software jako služeb, dostupných z libovolného zařízení, v libovolnou dobu, libovolném místě a které jsou plně programovatelné a přizpůsobitelné. K dosažení vize Microsoft dodává .NET platformu a .NET zkušenosti uživatelů, postavené na veřejných Internetových protokolech a standardech, společně s nástroji a službami, které spojují výpočty a komunikace novým produktivním způsobem. Platforma Microsoft .NET je jednoznačně navržena pro zrychlený vývoj aplikací, integraci a orchestraci libovolné skupiny webových XML služeb a aplikací do jediného řešení, a reprezentuje evoluci Windows DNA architektury.

PŘÍLOHA

Slovníček

Assembly: Je základním stavebním blokem programových komponent v .NET a představuje jednotku pro nasazení a řízení verzí. Assembly se může skládat z jedné DLL nebo může být složena z mnoha DLL a zdrojů v několika souborech. Assembly poskytuje odlišný pohled na aplikace napsané pro.NET Framework.
CLR: Common Language Runtime. Jde o prostředí pro běh aplikací, které poskytuje základní .NET služby. Při běhu aplikace nebo komponenty CLR řídí každou otázku jejího běhu, včetně řízení paměti, vynucení bezpečnostních pravidel, stejně jako splnění všech závislostí, které komponenta může mít vzhledem k jiným komponentám.
Cluster: Několik vzájemně propojených serverů za účelem podpory redundantnosti aplikací, které na nich běží. Pokud jeden z těchto serverů selže, ostatní automaticky přeberou jeho závazky (aplikace nebo funkcionalitu), dokud se server nedostane zpět do funkčního stavu.
HTTP: HyperText Transfer Protocol. Je to standardní protokol definující způsob, jakým jsou přenášena data mezi webovým serverem (jako je Microsoft Internet Information Server) a webovým prohlížečem (jako je Microsoft Internet Explorer).
Scale-Up (škálování do výšky): Přístup k vylepšení škálovatelnosti zahrnující přidávání nebo záměnu hardwarových komponent v serverech vedoucí ke zvýšení výkonnosti. Microsoft .NET Enterprise Server program obsahuje klíčovou spolupráci s výrobci hardware včetně IBM, Compaq, HP a Unisys, kteří poskytují nejlepší řešení pro škálování do výšky.
Scale-Out (škálování do šířky): Tento přístup vylepšuje škálovatelnost řešení formou dodání dalších serverů a vylepšuje celkový procesní výkon. Microsoft .NET Enterprise servery obsahují interní podporu pro Clustery serverů a webových farem, které vám umožňují postupné zvyšování procesního výkonu aplikací.
SOAP: Simple Object Access Protocol. Protokol definuje způsob, jakým aplikace (webové služby) v prostředí Internetu navzájem komunikují. SOAP dotaz obsahuje všechny informace potřebné pro vyvolání vzdálené procedury po Internetu. Je přenesen na server přes HTTP transportní protokol. SOAP dotazům je v převážné většině povolen průchod přes firewall, protože firewally jsou obecně konfigurovány k propouštění HTTP dotazů.
UDDI: Universal Description, Discovery and Integration. Jde o standardní specifikaci, která umožňuje společnostem centrálně zveřejnit informace o nabízených webových službách ve všeobecně dostupném Světovém Obchodním Registru. Obsahuje kontakty, produkty a služební informace o standardech a technologiích, které společnost používá při vedení online obchodu. Také nabízí vyhledávácí nástroj pro lokalizaci potenciálních prodejců specifických produktů nebo služeb a definuje pravidla jak firmy kontaktovat.
Web Farm (Webová farma): Několik webových serverů spolupracujících při realizaci škálovatelných webových řešení, postavených na technologii vyvažování výkonu. Proces vyvažování výkonu rovnoměrně rozkládá dotazy od množiny uživatelů mezi servery za účelem minimalizace přetížení jednoho serveru.
XML: eXtensible Markup Language. Jde o pružný, platformově nezávislý a standardizovaný způsob reprezentace dat přenášených mezi webovými servery nebo webovými službami.

Odkazy na zdroje

Následující seznam není úplný a vyčerpávající seznam článků, ale nasměruje vás na nejsrozumitelnější články o .NET.

Obchod
Hodnota .NET pro obchod: „Obchodní“ sekce na ***http://www.microsoft.com/net/***

IT Management
Windows Datacenter program:
http://www.microsoft.com/windows2000/guide/datacenter/overview/default.asp
.NET Enterprise servery:
http://www.microsoft.com/servery/net/default.asp

Technologie:
Microsoft SharePoint Portal Server:
http://www.microsoft.com/severy/sharepoint/default.asp
Vývoj Microsoft DNA:
http://www.microsoft.com/business/products/webplatform/default.asp
Sekce IT profesionálů:
http://www.microsoft.com/net/
XML Framework:
http://msdn.microsoft.com/xml/default.asp
Informace o COM:
http://www.microsoft.com/net/developer/framework_com.asp
UDDI:
http://www.uddi.org
http://www.microsoft.com/presspass/features/2000/sept00/09-06uddi.asp
MSDN White papers:
http://msdn.microsoft.com/library/default.asp?url=/library/techart/netanchor.htm

Obecné stránky
http://www.microsoft.com/net
http://msdn.microsoft.com/net
http://www.GotDotNet.com/

GotDotNet byl vytvořen pro podporu přímé diskuze a otevřené spolupráce mezi .NET vývojářskou komunitou a Microsoft .NET Framework týmem. GotDotNet je veřejným webovým portálem, kde .NET vývojáři mohou vystavit své příklady a získat informace o .NET Framework. Byl specificky navržen pro čistý a nefiltrovaném pohled dovnitř .NET Framework a tým uvnitř firmy Microsoft je zodpovědný za psaní, testování a řízení různých technologií, ze kterých je složen .NET. Z tohoto důvodu se liší od podobných portálů jako ***http://www.microsoft.com*** nebo ***http://msdn.microsoft.com***.

http://www.asp.net
http://commnet.pdc.mscorpevents.com
http://www.microsoft.com/servers/net/default.asp
http://www.ibuyspy.com

Kapitola 3:

Proč Microsoft .NET?

Abstrakt

Tato kapitola pojednává o technologiích, z nichž se .NET platforma sestává a výhodách, které přinášejí vývojářům.

ÚVOD

Microsoft .NET přináší pevný základ pro vytváření aplikací a poskytuje významné technologické výhody vývojářům. Obecná otázka vývojářů zní: „Proč Microsoft .NET?“.

Článek napomáhá odpovědět na tuto otázku a poskytuje základní informace o různých technologiích, které tvoří platformu .NET. Také diskutuje technické výhody realizace .NET řešení.

ARCHITEKTURA .NET FRAMEWORK

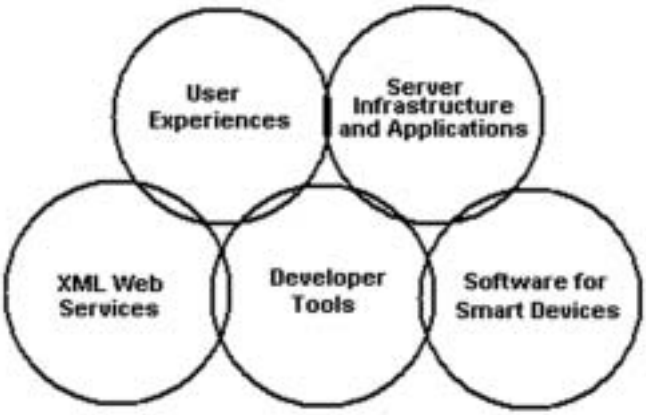
Internet jako aplikační infrastruktura

ARCHITEKTURA .NET FRAMEWORK

Microsoft .NET rozšiřuje koncepci Internetu a operačních systémů tím, že Internet samotný pokládá jako základ pro nové distribuované operační systémy, které podporují nový typ bohatší základní infrastruktury pro vytváření aplikací a služeb. Umožní vám vytvářet programy, které překonávají bariéry zařízení a učiní konektivitu do Internetu součástí vašich aplikací. Co více, je použitelný pro všechny typy aplikací a tak nebudete muset nadále přemýšlet o dvou různých oddělených infrastrukturách – jedné pro webové aplikace a jiné pro interní a desktopové aplikace.

Základní .NET komponenty

Z perspektivy lze základní komponenty .NET nakreslit následujícím způsobem:



Obrázek 9 Základní komponenty .NET

Zkušenosti uživatelů

Reprezentují způsob, jakým pracujete a manipulujete s personalizovanými informacemi, které vám jsou k dispozici formou připojených webových služeb a poskytovaných chytrými zařízeními.

- Osobní a integrované. .NET zkušenosti jsou směrovány na uživatele, na rozdíl od široké množiny informací získávaných od různých jiných poskytovatelů a integrují uživatelská data a nastavení do jedné aplikace.
- Připojené webové XML služby. Použitím HTTP, XML a SOAP může řada služeb, přizpůsobených potřebám individuálních uživatelů, podporovat jednu celkovou zkušenost.
- Chytrá zařízení. Tato zařízení ví mnoho o vás (a mohou používat vaši .NET identitu ve službách, kde je jí zapotřebí), o počítačové síti (včetně schopnosti pracovat jak odpojeném, tak připojeném režimu), o různých informacích (optimalizují, jak mají být tyto informace zobrazovány a jak se s nimi má pracovat), o ostatních zařízeních (včetně ostatních zařízení používaných jinými uživateli, které potřebujete kontaktovat, nebo s nimiž si vyměňujete data) a o službách (včetně toho, jaké služby jsou uživateli dostupné a jak se k nim připojit).

Serverová infrastruktura a aplikace
Serverová infrastruktura a aplikace jsou realizovány .NET Enterprise servery a umožňují vám vytvářet, nasazovat a provozovat Webové služby. Klíčové technologie obsahují podporu XML, škálování do šířky a orchestrace obchodních procesů. .NET Enterprise servery čítají následující Microsoft produkty:

- Internet Security and Acceleration Server
- Host Integration Server 2000
- SQL Server™ 2000
- Exchange Server 2000
- Application Center Server 2000
- BizTalk™ Server 2000
- Commerce Server 2000
- Mobile Information Server
- SharePoint™ Portal Server

Více informací o .NET Enterprise serverech získáte na ***http://www.microsoft.com/servers.***

Webové XML služby
Jsou to opakovatelně použitelné stavební bloky ve formě služeb přístupných přes Internet a v současné době zahrnují Passport.NET (pro uživatelskou autentikaci), služby pro ukládání souborů, řízení uživatelských nastavení, práci s kalendářem a mnoho dalších jakým je např. malý obchodní portál bCentral™ a MSN® síť internetových služeb. Tyto stavební bloky jsou zaměřeny na poskytování jednoduché, konkurenční a personifikované zkušenosti koncovým uživatelům, formou výměny smysluplných uživatelsky orientovaných dat mezi službami.

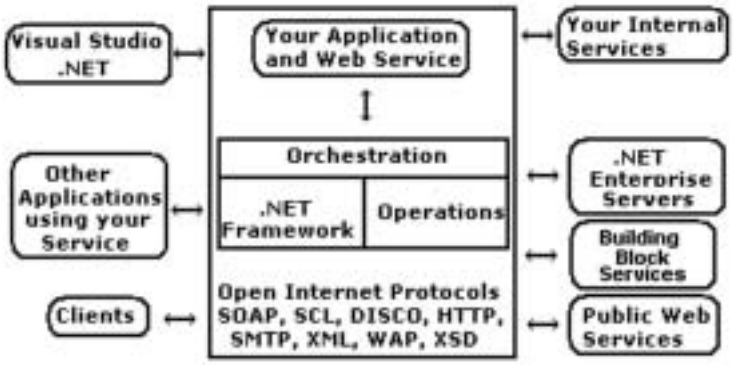
Vývojové nástroje
Jde o produkty a nástroje potřebné pro vytváření a provozování nové generace webových XML služeb a uživatelských zkušeností a obsahují Visual Basic.NET, C# a .NET SDK (Software Development Kit). Tyto nástroje vám umožňují jednodušeji vytvářet aplikace, které vystavují a konzumují webové služby a rozšiřují možnosti multi-jazykové tvorby programů s podporou různých zařízení.

Software pro chytrá zařízení
Zpřístupňuje chytrá zařízení pracující v Internetu, integruje služby .NET identity, práci v připojeném a odpojeném režimu, optimalizuje zobrazení informací, provádí spojení s ostatním zařízeními a zpřístupňuje XML webové služby uživatelům.

Zpřístupnění webových služeb

Rostoucí počet softwarových řešení je v současnosti navrženo ve složitých n-vrstvých systémech, které často spojují mnoho heterogenních aplikací provozovaných v rozsáhlých sítích.

Vstupujeme do další fáze výpočetní techniky, fáze zprostředkované Internetem. V tomto kontextu již software není něčím instalovaným z CD, ale předplacenou službou (jako je provedení telefonního hovoru nebo sledování placeného televizního kanálu v hotelu) pomocí komunikačního média. Tento nový typ software doplňuje úzce vázané, vysoce produktivní aspekty n-vrstvých výpočetních systémů volně vázanými, událostmi řízenými koncepcemi Webu a používá Internetové XML standardy. Nový styl výpočtů je realizován webovými XML službami. Webová XML služba je softwarová komponenta nebo aplikace, která programově zpřístupňuje po Internetu nebo Intranetu své vlastnosti formou standardních internetových protokolů jako Simple Object Access Protocol (SOAP) pro vzájemnou komunikaci programů, a XML pro datovou reprezentaci. Zjednodušeně lze o nich uvažovat jako o komponentově orientovaném programování na Webu. Model webových služeb je ilustrován na obrázku 10.



Obrázek 10 Model webových služeb

Koncepčně integrujete externí webové služby do svých aplikací směrováním požadavků přes Internet do služeb na vzdálených systémech. Např. služby jako Microsoft Passport mohou vašim aplikacím poskytnout autentikaci. Programováním služeb Passportu můžete získat výhodu jeho infrastruktury, spolehnout se na údržbu databáze uživatelů Passportu a mít jistotu, že má dostatečnou výkonnost, pravidelné zálohování a podobně.

Distribuce a integrace aplikační logiky v Internetu

Koncepce distribuce aplikační logiky v počítačové síti není nová. Co je nového, je koncepce distribuce aplikační logiky v Internetu.

Distribuovaná aplikační logika dříve volala distribuované objektové modely jako Distributed Component Object Model (DCOM) firmy Microsoft, Component Request Broker Architecture (CORBA) skupiny Object Management Group nebo Remote Method Invocation (RMI) firmy Sun. Při použití jednoho z uvedených systémů lze zachovat mnoho z bohatých a precizních postupů, na které jste zvyklí u lokálního programovacího modelu a přitom umisťovat služby na vzdálené systémy.

Problémem uvedených systémů je, že špatně škálují v Internetu. Jejich závislost na úzké vazbě mezi konzumentem služby a službou samotnou vyžaduje homogenní infrastrukturu a často znamená jejich křehkost. Pokud se změní implementace jedné ze stran, ostatní strany přestanou fungovat. Ve skutečnosti není nic špatného na požadavku úzce vázané infrastruktury a mnoho aplikací bylo na ni úspěšně postaveno. Model však jednoznačně nebude moci růst. Jakmile společnosti pohlcují jedna druhou, dodavatelé informačních technologií končí své podnikání nebo mění své modely, je velice těžké garantovat jednotnou a unifikovanou infrastrukturu. Neexistuje žádná záruka, že služba, s níž potřebujete komunikovat na vzdáleném konci vašeho spojení, bude mít požadovanou infrastrukturu. Nemusíte dokonce ani vědět, jaký operační systém, objektový nebo programový model používá.

Standardní formát popisu dat pro webové služby

Webové služby jsou, podle definice, volně vázáné. Znamená to, že můžete změnit implementaci na libovolném z konců spojení bez ovlivnění druhého. Technicky je vlastnosti dosaženo pomocí technologie zpráv, asynchronních volání pro robustní výkonnost, webových protokolů jako HTTP a ze všeho nejdůležitěji XML, pro docílení celosvětového záběru.

Klíč k realizaci webových služeb fungujících napříč Internetem a jeho heterogenní infrastrukturou spočívá v dohodnutí jednoduchého formátu pro popis dat. Tímto formátem je XML. Specificky, webové služby potřebují XML k vyřešení čtyř odlišných problémů:

- Definuje nízko úrovňový formát pro komunikaci mezi procesy. Systémy potřebují na nejnižší úrovni komunikovat společným jazykem. Komunikující aplikace konkrétně potřebují sadu pravidel pro reprezentaci různých datových typů (např. celá čísla a pole) a pravidla pro reprezentaci příkazů (tzn. co má být prováděno s daty). Aplikace také potřebují způsob, jakým rozšiřovat tento jazyk, pokud je nutné. Simple Object Access Protocol (SOAP) reprezentuje obecnou množinu pravidel pro reprezentaci dat a příkazů a jejich rozšiřování.
- Popis služeb. Jakmile aplikace disponují obecnými pravidly pro popis reprezentace datových typů a příkazů, potřebují metodu, jak popsat konkrétní sadu dat a příkazů, které přijímají. Nestačí například prohlásit, že aplikace přijímá celá čísla. Musí existovat nějaký způsob, jak deterministicky říci, že při předání dvou celých čísel dojde k jejich vynásobení. Web Service Description Language (WSDL) je XML gramatikou, kterou mohou vývojáři a vývojové nástroje použít pro popis schopností webové služby.
- Vyhledání webových služeb. Určitě budete potřebovat sadu pravidel pro lokalizaci popisu služeb. Jak člověk nebo nástroj nalezne popis schopností webových služeb? Discovery protokol (DISCO) poskytuje sadu pravidel umožňujících člověku nebo vývojovým nástrojům automaticky vyhledávat popis služeb ve formátu WSDL.
- Centrální registrace webových služeb. Iniciativa Universal Description, Discovery and Integration (UDDI) se zaměřuje na poskytování kompletního seznamu firem působících v online světě, pomocí seznamu služeb, které nabízejí. Obchodníci se mohou bez jakýchkoli nákladů registrovat v UDDI Universal Business Registry formou podrobných kontaktních informací společně s produkty a službami, které nabízejí. Kupující pak budou schopni listovat adresářem (opět zdarma) a nacházet společnosti poskytující produkty nebo služby, které požadují.

Jakmile jsou tyto vrstvy vybudovány, budete mít možnost vyhledávat webové služby, vytvářet jejich instance ve formě objektů a integrovat je do svých aplikací.

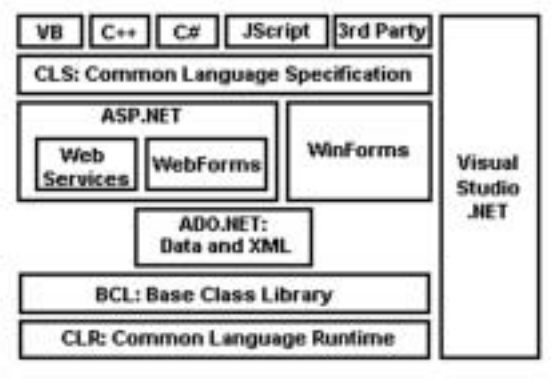
Používání otevřených průmyslových standardů

Všechny příbuzné technologie webových služeb jsou průmyslovými standardy s velmi širokou podporou a nejsou proprietární pro firmu Microsoft. Následující seznam představuje další zdroje vztahující se k těmto technologiím.

- Více informací o XML najdete na:
  http://www.w3.org/XML
  http://msdn.microsoft.com/xml
- Více informací o WSDL najdete na:
  http://msdn.microsoft.com/xml/general/wsdl.asp
- Více informací o SOAP najdete na:
  http://www.w3.org/TR/SOAP
  http://msdn.microsoft.com/soap
- Více informací o SOAP najdete na:
  http://msdn.microsoft.com/xml/general/disco.asp
- Více informací o UDDI najdete na:
  http://www.uddi.org
  http://www.microsoft.com/PressPass/features/2000/sept00/09-06uddi.asp

Upravená a moderní technologická architektura

Obrázek 11 ukazuje základní architekturu .NET Framework.



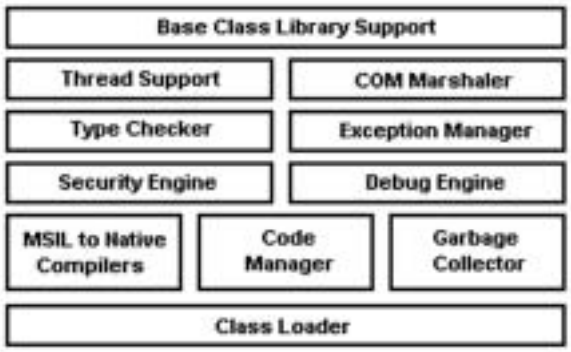
Obrázek 11 Microsoft .NET Framework

Základní komponenty .NET Framework jsou následující:

- Common Language Runtime (CLR). CLR je exekučním prostředím pro běh .NET Framework aplikací. Poskytuje mnoho služeb, včetně zavedení a spuštění kódu, izolaci aplikační paměti, správu paměti, zpracování výjimek, přístup k metadatům (rozšířené typové informace) a konverzi mezikódu (Microsoft Intermediate Language neboli MSIL) na nativní kód platformy.
- Base Class Library (BCL). BCL poskytuje rozsáhlou množinu tříd, logicky setříděných do hierarchických jmenných prostorů, které umožňují přístup k interním vlastnostem operačního systému.
- ADO.NET. Jde o novou generaci technologie přístupu k datům ActiveX® Data Object (ADO), obsahující významná vylepšení zaměřená na odpojenou podstatu Webu.
- ASP.NET. Jde o pokročilou verzi Active Server Pages (ASP) určených pro vývoj webových aplikací (nasazením webových formulářů) a webových služeb.
- Common Language Specification (CLS). Specifikace je zodpovědná za zpřístupnění většiny z výše zmíněných technologií všem jazykům, které platforma .NET podporuje. CLS sama o sobě není technologií a neexistuje pro ni žádný zdrojový kód. Definuje sadu pravidel, která realizují kontrakty řídící vzájemnou spolupráci mezi kompilátory jazyků a knihovnami.
- Windows Forms. Tento programový model a sada ovládacích prvků nabízí robustní architekturu pro vývoj Windows aplikací.
- Visual Studio.NET. Poskytuje nástroj, který vám umožní využít vlastnosti .NET Framework při vývoji konkrétních aplikací.

CLR zjednodušuje vývoj

Common Language Runtime (CLR) hraje roli jak při vývoji, tak za běhu .NET aplikací. Za běhu je CLR odpovědný za správu exekučního prostředí .NET (tzv. řízeného) kódu, včetně alokace paměti a řízení pracovních vláken, stejně jako vynucení bezpečnostních pravidel. Jelikož CLR automaticky provádí tyto vývojářské úkoly, dochází ke zkrácení doby vývoje produktu a ostatní úkoly vývoje jsou zjednodušeny. Např., automatický garbage collection znamená, že se již nikdy nebudeme muset starat o úniky paměti. Některé z důležitých komponent CLR jsou na obrázku 12.



Obrázek 12 Komponenty CLR

Framework je multi-jazykový a rozšiřitelný

Prostředí pro běh aplikací nejsou ničím novým v oblasti programovacích jazyků. Základní úlohou exekučního prostředí v .NET Framework (a co jej skutečně odlišuje) je realizace univerzálního společného pro všechny programovací jazyky.

Základní třídy .NET Framework (realizované ve formě Base Class Library, neboli BCL) poskytují univerzální, objektově orientovanou, hierarchickou a rozšiřitelnou množinu knihovních tříd používaných při vývoji. V současnosti můžete jako vývojář v jazyce C++ pracovat s Microsoft Foundation Classes (MFC), jako Java vývojář by jste použil Windows Foundation Classes (WFC) a jako programátor ve Visual Basic® přistupujete k Visual Basic API.

.NET Framework unifikuje neslučitelná programová prostředí, s nimiž musíte v současnosti pracovat. Při výkonu své profese se ve výsledném efektu nebudete muset učit pracovat s několika odlišnými prostředími. Jako vývojář komponent se nemusíte rozhodovat v jakém jazyce programovat, neboť vždy budete psát pro CLR a vědět, že vaše práce bude dostupná ze všech programovacích jazyků.

Spolupráce napříč jazyky

Mezi mnohé klady .NET patří jazyková transparentnost. Při výběru libovolného řízeného jazyka (managed language) máte identický přístup k celé platformě. Tím, že existuje společná množina API napříč všemi programovacími jazyky, .NET Framework podporuje dědičnost, zpracování chyb a ladění mezi jednotlivými jazyky. Jako vývojáři můžete využít výhody .NET platformy z libovolného jazyka, který jste si zvolili.

Multiplatformní návrh

.NET představuje nový způsob vývoje software pro Windows a případně i jiné platformy. .NET není neoddělitelně spjat s operačním systémem Windows. Například, mezikód (MSIL) generovaný překladači .NET jazyků není specifický pro konkrétní procesor a BCL třídy jsou navrženy tak, aby pracovaly s libovolným operačním systémem.

.NET umožňuje aplikacím jednoduché šíření na různé platformy díky svým schopnostem spolupráce a rozšiřitelnosti. Všechny .NET aplikace dodržují CLI (Common Language Infrastructure). CLI se skládá z CLR (Common Language Runtime) a BCL (Base Class Library). Jak CLI, tak jazyk C# (vysloveno „sí šarp“; moderní komponentově orientovaný jazyk) byly předány mezinárodní standardizační komisi ECMA (European Computer Manufacturer's Association) k mezinárodnímu schválení (další informace naleznete na ***http://www.microsoft.com/PressPass/press/2000/Nov00/VSFWbeta1PR.asp).*** Microsoft se zavázal udržovat jejich verze na základě ECMA standardů.

BCL nahrazuje API

.NET Framework poskytuje společnou sadu funkcionality a služeb ve formě Base Class Library (BCL). Tato knihovna nabízí náhradu za mnoho současných Win32® API, což znamená, že jako vývojář budete psát kód pro konkrétní platformu místo konkrétního operačního systému. Zatímco Win32 API jsou stále dostupná prostřednictvím Frameworku, ve většině případů si budete moci zvolit třídy z BCL. Tím, že je kód organizován do jasně definovaných jmenných prostorů, je jednodušší nalézt třídu pro realizaci konkrétní úlohy.

Řízený kód a Assembly

Veškeré .NET jazyky jsou určeny pro platformu CLR a generují „řízený kód“ („Manged code“). Řízený kód je zabalen, nasazen a opatřen verzí pomocí tzv. assembly. Otázka assembly je diskutována dále v tomto dokumentu v kapitole „Možnosti nasazení“.

Významné vlastnosti realizované v .NET Framework

V následujících bodech jsou uvedeny významné vlastnosti .NET Framework:

- Konzistentní programový model: Všechny aplikační služby jsou dostupné pomocí společného objektově orientovaného programového modelu, na rozdíl od některých vlastností operačních systémů v současnosti dostupných formou DLL funkcí a jiných vlastností dostupných pomocí COM objektů.
- Zjednodušený programový model: .NET se snaží významně zjednodušit infrastrukturu a tajemné konstrukce nutné ve Win32 a COM. Konkrétně, vývojáři se již nemusí nadále starat o to co je v Registry, co to jsou GUID a COM rozhraní IUnknown, jaký je mechanismus počítání referencí a metody AddRef a Release, HRESULT atd. Tyto koncepce v .NET prostě vůbec neexistují.
- Spusť jednou, spusť pokaždé. Všichni vývojáři dobře znají problém DLL Hell. Během instalace komponent nové aplikace může dojít k přepsání komponent aplikace starší a tato aplikace může vykazovat podivné chování nebo přestat fungovat úplně. .NET architektura nyní odděluje aplikační komponenty a tak aplikace nahrává pouze ty komponenty, se kterými byla vytvořena a proti kterým byla testována. Jestli aplikace běží po instalaci, pak by aplikace měla běžet vždy.

- Spouštění na více platformách: Jakmile je řízená .NET aplikace napsána a přeložena do MSIL, může běžet na libovolné platformě, která podporuje .NET CLR (včetně jiných platforem než je Windows). Hodnotu tohoto exekučního modelu oceníte ihned, jakmile budete potřebovat podporu různých hardwarových konfigurací a operačních systémů.
- Jazyková integrace: Zatímco COM umožňuje jednotlivým programovacím jazykům vzájemnou spolupráci, pak .NET umožňuje jazykům, aby byly integrovány jeden s druhým. Je například možné vytvořit jednu třídu v C# a odvodit z ní třídu implementovanou v jazyce Visual Basic.NET. .NET dosáhl uvedené schopnosti díky definice a realizaci Common Type System (CTS), společného pro všechny .NET jazyky. Microsoft poskytuje několik kompilátorů, které produkují kód pro .NET CLR: C++ s řízenými rozšířeními, C#, Visual Basic.NET (který nyní nahrazuje Visual Basic Scripting Edition a Visual Basic for Applications (VBA)), JScript® a J#. Vedle firmy Microsoft i řada jiných společností produkuje překladače dalších jazyků pro platformu .NET CLR.
- Opakované použití kódu: Za použití výše uvedeného mechanismu můžete vytvářet své vlastní třídy, které nabízejí služby aplikacím třetích stran. To samozřejmě významně zjednodušuje opakované použití kódu a rozšiřuje trh výrobcům komponent.
- Automatická správa zdrojů pomocí garbage collection: Programování vyžaduje značné znalosti a disciplínu. Zejména to platí v případě, kdy pracujeme se zdroji jako jsou soubory, paměť, síťová spojení, databázové zdroje atd. Jedna z nejčastěji se vyskytujících chyb nastává, když aplikace zanedbá navrácení některého ze zdrojů a způsobí, že tato aplikace nebo jiná havaruje. .NET CLR automaticky sleduje používání zdrojů a garantuje, že vašim aplikacím zdroje nikdy neuniknou. Procesu se říká „garbage collection“. Neexistuje žádný způsob, jak explicitně uvolnit paměťové zdroje, neboť jsou vždy automaticky sesbírány v chvíli, kdy již nejsou potřeba.
- Typová bezpečnost: .NET CLR může verifikovat bezpečnost veškerého vašeho kódu. Typová bezpečnost zajišťuje, že k alokovaným objektům je vždy přistupováno kompatibilním způsobem. Pokud je například vstupní parametr deklarován jako 4-bajtová hodnota, pak CLR detekuje a zachytává pokusy o předání parametru s 8-mi bajtovou hodnotou. Obdobně, pokud objekt zabírá v paměti 10 bajtů, aplikace si nemůže vynutit přečtení více než 10 bajtů. Typová bezpečnost také znamená, že instrukční tok smí být přesměrován pouze na předem známé pozice (jmenovitě vstupní body metod). Není zde možné vytvořit libovolný odkaz do paměti a způsobit přenesení instrukčního toku na toto místo. Typová bezpečnost a nedostupnost přímých operací s ukazateli jsou základem bezpečnostního modelu a eliminují mnoho častých programátorských chyb a klasických útoků na systémy, jako je zneužití přetečení vyrovnávací paměti.
- Bohatá podpora pro ladění: Protože .NET CLR je společným základem pro mnoho jazyků, je nyní mnohem jednodušší používat komponenty v aplikacích. Komponenty mohou být napsány v jazycích, které jsou pro ně nejvhodnější. .NET CLR plně podporuje ladění napříč komponentami napsanými v různých jazycích. Můžete například krokovat dovnitř kódu ovládacího prvku napsaného v jazyce C# z klientského programu napsaného v jazyce Visual Basic.NET.
- Konzistentní zpracování chyb: Jednou z nejvíce frustrujících skutečností programování na Windows platformě je nejednotný způsob, jakým jsou hlášeny chyby. Některé z funkcí vrací Win32 chybové kódy, jiné vrací HRESULT a další vyhazují výjimky. V .NET jsou všechny chyby hlášeny pomocí výjimek. Normální běh programu se po vyvolání výjimky zastaví a je vyhledáván odpovídající ovladač výjimky. Výjimky vám také dovolí izolovat kód

pro zpracování chyb od zbytku aplikační logiky. Tímto se výrazně zjednoduší vytváření, čitelnost a údržba kódu. Výjimky navíc fungují přes hranice modulů a programovacích jazyků.

- Nasazení: V současné době může být velice složité instalovat a nasazovat Windows aplikace. Sestávají se většinou z více souborů, nastavení Registry a zástupců, kteří mají být vytvořeni. Navíc je téměř nemožné úplně aplikace odinstalovat. Společně s Windows 2000 Microsoft uvedl Windows Installer, nový instalační nástroj, který pomáhá řešit tyto problémy, avšak není vyloučeno, že firma může vytvořit instalační balíček pro Windows Installer jenž může úplně selhat. .NET se snaží o úplnou eliminaci těchto problémů. .NET komponenty nejsou registrovány v Registry a instalace většiny .NET aplikací nevyžaduje nic více než překopírování souborů do adresáře. Deinstalace .NET aplikace je pak jednoduchá jako smazání těchto souborů. CLR také umožňuje souběžné (side-by-side) nasazení, kdy můžeme mít dvě verze stejné aplikace běžící ve stejnou chvíli na stejném počítači.
- Bezpečnost: Tradiční operační systémy realizují izolaci a řízení přístupu na základě uživatelských účtů. Tento přístup byl ověřen jako užitečný model, avšak přepokládá, že všechen kód je identicky důvěryhodný. Předpoklad byl splněn když všechen kód byl instalován z fyzických médií (jako je CD-ROM) nebo důvěryhodných firemních serverů. Se vzrůstající závislostí na mobilním kódu, jako jsou webové skripty, stahování aplikací z Internetu a přílohy elektronické pošty, vyvstává potřeba mnohem jemnějšího řízení chování aplikací. Takovou kontrolu zprostředkovává .NET Code Access Security model. Aplikace si často potřebují vynutit funkcionalitu založenou na koncepci rolí, odlišných od individuálních účtů. Teorie rolí byla poprvé uvedena v Microsoft Transaction Server (MTS) a dále rozšířena jako součást COM+. .NET podporuje definice aplikačních rolí a řízení přístupu v závislosti na těchto rolích. Uvedené mechanismy jsou rozšířeny takovým způsobem, aby obecně vyhovovaly internetovým a heterogenním prostředím.

MODERNÍ TECHNOLOGIE PŘEKLADAČŮ

MSIL činí váš kód nezávislý na CPU a operačním systému

Všechny překladače .NET jazyků generují Microsoft Intermediate Language (MSIL) pro platformu CLR. MSIL jazyk vytvořený firmou Microsoft na základě konzultací s mnoha externími a akademickými tvůrci jazyků a překladačů je nezávislý na typu procesoru. MSIL je daleko vysokoúrovňovějším jazykem než je většina strojových jazyků procesorů. Rozumí objektovým typům a obsahuje instrukce, které zakládají a inicializují objekty, volají virtuální metody objektů a přímo manipulují s elementy polí. Má dokonce instrukce pro vyvolání a zachycení výjimek při zpracování chyb.

Důležité je zmínit skutečnost, že MSIL není vázán na některou konkrétní procesorovou platformu. To znamená, že PE (Portable Executable) soubor, obsahující MSIL instrukce, může běžet na libovolné procesorové platformě, pokud operační systém na této platformě obsahuje .NET CLR. Počáteční plány firmy Microsoft byly orientovány na implementaci .NET CLR pro operační systémy Windows 98, Windows NT 4.0 a Windows 2000. V současné době je tento seznam rozšířen o systémy Windows Me a Windows CE.

Další výhodou MSIL je realizace abstraktní hardwarové vrstvy. Zatímco první dostupná verze CLR běží pouze na platformě 32-bitových Windows, při vývoji aplikací s MSIL umožňuje větší nezávislost na základním operačním systému. MSIL významně napomůže portaci kódu na 64-bitovou verzi Windows ve chvíli, kdy na této platformě bude CLR dostupný.

Nakonec, MSIL je naprosto nezbytným pro realizaci .NET bezpečnosti, jelikož verifikace (jak byla diskutována dříve) je schopna zkoumat úmysly kódu nezávisle na programovacím jazyce použitém při vývoji.

MSIL je řízen pomocí CLR

MSIL instrukce nemohou být prováděny na současných procesorech a tak výkonné jádro CLR musí nejdříve přeložit MSIL instrukce do nativních instrukcí procesoru. CLR poskytuje MSIL přístup k vlastnostem .NET Framework.

Potřeba překládat MSIL do nativního kódu vzbuzuje otázku, zda má CLR převádět všechen MSIL kód aplikace na CPU instrukce v době jejího zavádění. Výhodou takového přístupu je, že program běží velice rychle. Nevýhodou však je, že překlad MSIL vyžaduje svůj čas, který významně prodlužuje start aplikace. Navíc je velmi vzácné, že uživatel využije veškerý kód za běhu aplikace. Pokud CLR přeloží všechen MSIL kód na CPU instrukce, existuje vysoká pravděpodobnost, že bude promrháno mnoho času a paměti. CLR řeší tento problém pomocí tzv. Just-in-Time překladu (překladu na požádání).

Optimalizovaný Just-in-time překlad

CLR překládá MSIL instrukce, jakmile jsou funkce volány. Když CLR nahraje typ reprezentovaný třídou, připojí ke každé jeho metodě náhradní kód. Ve chvíli volání metody náhradní kód přesměruje běh programu do komponenty CLR, která je odpovědná za překlad MSIL kódu metody do nativního kódu. Jelikož je MSIL překládán za běhu (Just-in-Time, JIT), této komponentě se velice často říká JIT překladač. Jakmile JIT překladač přeložil MSIL, náhradní kód metody je vyměněn za adresu přeložené kódu. Kdykoli v budoucnosti je tato metoda volána, přímo se

spustí nativní kód a JIT překladač se již do procesu nezapojí. Zajisté si představíte, že dochází k významnému nárůstu výkonnosti ve srovnání se standardním runtime interpretem. JIT překladače navíc podporují „zahození“ kódu. Zahození kódu je schopnost CLR odstranit z paměti nepoužívané bloky kódu a uvolnit tak paměť, která nebyla po jistou dobu běhu aplikace používána. Když CLR zahodí část kódu, samozřejmě nahradí metodu zástupným kódem, který JIT překladač spustí při příští snaze o vyvolání této metody.

Když JIT překladač kompiluje MSIL kód za běhu aplikace, ví mnohem více o exekučním prostředí než je známo v době běžného překladu. JIT kompilátor například může detekovat typ přítomného CPU jako Pentium III a generovat instrukce, které využijí výhod a libovolných výkonnostních rozšíření, jimiž Intel odlišuje Pentium III od Pentia II nebo Pentia.

JIT překladač navíc může generovat kód používající CPU registry, kterých by se běžný překladač vyvaroval. JIT překlad může vyústit ve schopný malý a rychlý kód, který pracuje jen díky předpokladům vytvořených za běhu programu.

Nárůst výkonnosti dopředním přeložením kódu

Další vlastnost nazvaná PreJIT poskytuje způsob, jak přeložit celou assembly do nativního kódu a ve chvíli prvního zpuštění uložit výsledek na disk. Když je assembly nahrána podruhé, je zavedena uložená, dopředu přeložená verze. Ve výsledku se aplikace spouští mnohem rychleji.

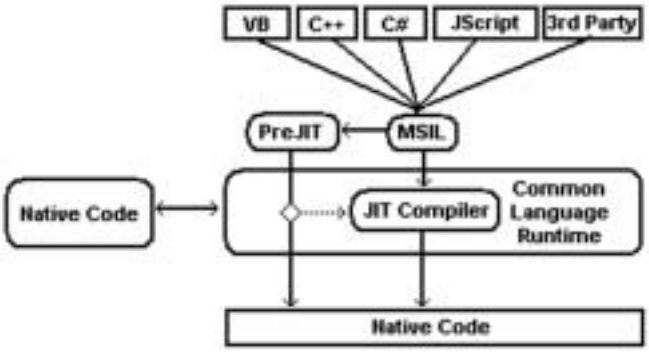
Jakmile si nainstalujete .NET SDK na svůj počítač, automaticky dochází na počátku k dopřednímu překladu několika systémových assembly jako je MSCORELIB, obsahující mnoho systémových tříd vyžadovaných každou .NET aplikací. Proces analyzuje MSIL binární kód a metadata assembly, počítá informace o struktuře třídy a překládá metody do jejich ekvivalentního nativního kódu.

Proces dopředního překladu se týká mnohem více systémových assembly. Dokonce máme možnost si takto přeložit i své vlastní assembly s pomocí nástroje příkazové řádky, nazvaného NGEN (Native Code Generator).

Po spuštění NGEN nad konkrétní assembly dojde k uložení výsledku do tzv. Global Assembly Cache (GAC) na lokálním pevném disku. Použitím techniky dopředního překladu assembly dosáhneme nižšího času potřebného pro zavedení assembly a vyšší výkonnost za běhu aplikace, protože není třeba JIT-překládat každou metodu při jejím volání z MSIL do nativního kódu.

Dynamické nahrazení komponent

NGEN zaznamenává informace o konfiguraci ve výstupním přeloženém souboru. Pokud se změní detaily v konfiguraci mezi dobou, kdy byl NGEN spuštěn a chvílí, kdy aplikaci spustíte (např. po instalaci nové verze jedné ze závislých assembly nebo po změně bezpečnostních pravidel), pak CLR automaticky detekuje tyto změny a vrátí se zpět k použití běžného JIT překladače. Tímto způsobem můžete zaměňovat assembly novými verzemi za běhu programu, bez problémů způsobených uzamknutím DLL knihoven. Přehled procesu JIT překladu je ilustrován na obrázku 13.

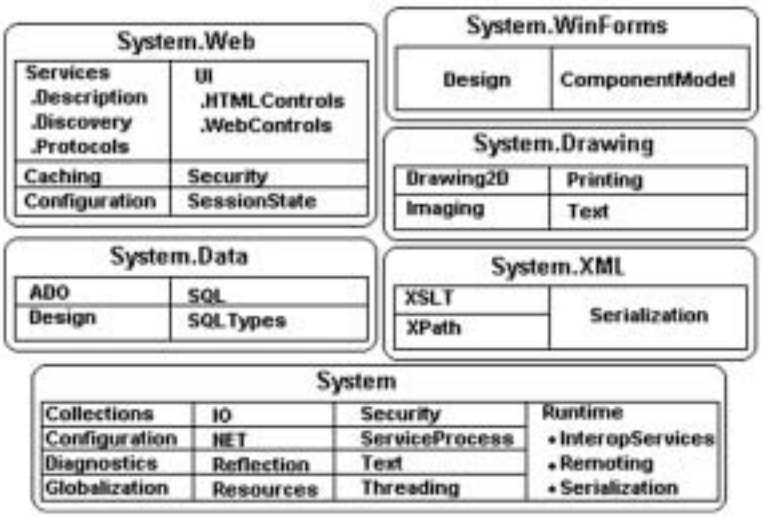


Obrázek 13 Překladače generují MSIL, který lze přeložit během instalace (pomocí PreJIT) nebo JIT překladačem za běhu. CLR detekuje libovolné změny vyžadující překlad.

Jmenné prostory realizují jednotné programové paradigma

Base Class Library (BCL) je součástí .NET jako základní framework assembly a obsahuje mnoho definic tříd, kde každá z nich zpřístupňuje jistou vlastnost níže položené platformy. Jelikož Microsoft definuje v BCL stovky tříd, knihovna je rozdělena do jmenných prostorů, které logicky sdružují příbuzné třídy.

Například, jmenný prostor System obsahuje základní třídu Objekt, z níž jsou bez výjimky odvozeny všechny ostatní třídy. Jmenný prostor navíc obsahuje třídy pro práci s výjimkami, správou paměti (garbage collection), konzolový v/v, stejně jako mnoho různých pomocných tříd poskytujících typové konverze, formátování datových typů, generování náhodných čísel a provádění matematických operací.



Obrázek 14 Jmenné prostory Base Class Library

Aby jste mohli pracovat s některou z uvedených vlastností, musíte použít odpovídající třídu definovanou v těchto jmenných prostorech. Potřebujete-li přizpůsobit chování libovolné z dostupných tříd, jednoduše odvoďte na jejím základě svou vlastní třídu.

.NET Framework je rozšiřitelný

.NET představuje vývojářům software jednotné programové paradigma díky objektově orientované podstatě platformy. Vývojáři mohou zakládat své vlastní jmenné prostory obsahující jejich vlastní třídy. Bezproblémově je lze spojit s existujícími třídami a významně tak zjednodušit vývoj software ve srovnání s klasickými programovacími přístupy ve Windows.

CLR implementuje společný typový systém

SPOLEČNÝ TYPOVÝ SYSTÉM

Formální specifikace typového systému implementovaného uvnitř CLR je nazvána Common Type System (CTS). CTS specifikuje, jak jsou definovány objektové třídy (nazvané typy). CTS například dovoluje třídě, aby obsahovala nula a více členů. Dostupné členy jsou:

- Položka: Datová proměnná, která je součástí interního stavu objektu. Položky jsou určeny svým jménem a typem.
- Metoda: Funkce, která vykonává operace nad objektem, většinou mění stav objektu. Metody mají jméno, podpis a modifikátory. Podpis určuje volací konvenci, počet parametrů (a jejich pořadí), typy parametrů a typ návratové hodnoty metody. Modifikátory mohou určovat, zda je metoda veřejná, soukromá, statická atd.
- Vlastnost: Vlastnosti obvykle vypadaly velice podobně jako položky a byly simulovány pomocí volání COM metod. Jazyky jako Visual C++ běžně zveřejňovaly interní mechanismy (například pomocí atributů Interface Definition Language), zatím co vyšší programovací jazyky jako Visual Basic uchovávaly tyto mechanismy skryté. Vlastnosti jsou velmi užitečné, neboť vývojářům dovolují vyčíslovat hodnoty pouze tehdy, když je to nutné a uživatelům třídy přistupovat k těmto hodnotám značně jednoduchou syntaxí. Vlastnosti se chovají jako „chytré položky“ a umožňují specifické zpracování ve chvíli, kdy jsou čteny nebo nastavovány. Vlastnosti .Net tříd jsou nyní prvořadými hráči a jsou definovány pomocí speciální jazykové syntaxe. Následující ukázka C# kódu například ilustruje třídu nazvanou Button, zpřístupňující vlastnost nazvanou Caption.

```
       class Button
       {
public string Caption // vlastnost
{
           get { return caption; }
  set { caption = value; }
  }
  private string caption;    //položka
  }
```

- Událost: Události realizují informační mechanismus mezi objektem a jinými zainteresovanými objekty. Např. tlačítko může nabídnout událost, která informuje jiné objekty o jeho stisknutí.

Pružné hranice viditelnosti objektů

CTS taktéž definuje pravidla určující viditelnost typů a způsob přístupu ke členům typů. Typy jsou buď viditelné vně assembly (tzn. klientům assembly), nebo jsou viditelné pouze pro kód obsažený uvnitř assembly samotné. Tím, že označíme typ jako veřejný (public) stává se viditelným vně assembly (je exportován). CTS tak stanovuje pravidla, pomocí kterých jsou definovány hranice viditelnosti (tzv. rozsah) typu a jejich metod. Common Language Runtime (CLR) následně hlídá tato pravidla viditelnosti. Nezávisle na tom, zda je uživateli typ viditelný, typy samy řídí, jestli volající má přístup ke jeho členům. Platné volby pro řízení přístupu k metodám nebo položkám jsou:

Typ přístupu	Popis
Public	Metoda je přístupná z libovolného místa v assembly.
Private	Metoda je přístupná pouze pro jiné metody stejné třídy.
Family	Metoda je přístupná z odvozených tříd, nezávisle na tom, jestli se nachází ve stejné assembly nebo ne.
Assembly	Metoda je přístupná z libovolného kódu ve stejné assembly.
Family a Assembly	Metoda je přístupná z odvozených typů, pokud jsou tyto typy definovány ve stejné assembly.
Family nebo Assembly	Metoda je přístupná z odvozených typů v libovolné assembly. Metoda je také přístupná z jiných typů ve stejné assembly.

Konzistentní chování objektů

CTS rovněž definuje pravidla řídicí chování objektů, včetně dědičnosti typů, virtuálních funkcí a životnosti objektů. Tato pravidla byla navržena tak, aby se přizpůsobila sémantice vyjádřitelné v současných jazycích. Ve skutečnosti se nemusíte učit pravidla CTS samotná, neboť jazyk, který si zvolíte, disponuje svou vlastní dobře známou syntaxí a typovými pravidly a mapuje tuto jazykově specifickou syntaxi na CTS syntaxi umístěnou v MSIL souboru. C# například používá modifikátor „protected“ pro reprezentaci přístupového typy Family.

Nejlepší je uvažovat o jazyce a chování vašeho kódu jako o dvou oddělených a odlišných věcech. Pomocí Visual C++ můžete definovat svou vlastní třídu s jejími vlastními členy. Samozřejmě jste mohli použít C# nebo Visual Basic.NET k definici stejné třídy se stejnými členy. Zatímco syntaxe použitá při definici této třídy se liší v závislosti na zvoleném jazyce, vlastnosti třídy jsou absolutně identické, protože chování této třídy definuje CTS CLR.

Jednoduchá dědičnost tříd, násobná dědičnost rozhraní

CTS podporuje jednoduchou implementační dědičnost a násobnou dědičnost rozhraní. CTS si také vynucuje, že všechny třídy musí (bez výjimky) být odvozeny od předdefinované třídy nazvané System.Object (kde Object je typové jméno uvnitř jmenného prostoru System). Object je kořenem všech ostatních typů a tedy garantuje, že každý typ má minimální množinu funkcionality. Typ System.Object konkrétně:

- umožňuje porovnání dvou objektů na rovnost
- umožňuje identifikovat objekt hash hodnotou
- umožňuje zjistit typ objektu za běhu aplikace
- umožňuje provést mělkou (bitově orientovanou) kopii objektu
- umožňuje získat řetězec reprezentující aktuální stav objektu

Multijazyková integrace a spolupráce

COM umožňuje vzájemnou komunikaci objektů vytvořených v různých jazycích pouze přes COM rozhraní. .NET CLR naproti tomu integruje všechny jazyky a chápe objekty vytvořené v různých jazycích jako sobě rovné a použitelné uvnitř kódu napsaného v úplně odlišném jazyce. CLR toho dosahuje díky standardní množině typů, samopopisných typových informací (nazvaných metadata) a společnému exekučnímu prostředí.

CLS jako podmnožina CTS

Pokud máte zájem vytvářet typy, které jsou jednoduše přístupné v jiných programovacích jazycích, je důležité, aby byly použity pouze ty vlastnosti zvoleného programovacího jazyka dostupné všem ostatním. Pro ulehčení problému, Common Language Specification (CLS) informuje výrobce překladačů o minimální množině vlastností, které jejich překladače musí podporovat, aby mohly využít CLR. Několik výrobců již dnes pracuje na .NET kompatibilních verzích svých překladačů.

Všimněte si, že CTS podporuje více vlastností než podmnožina definovaná v CLS. Pokud vás tedy nezajímá mezi-jazyková spolupráce, můžete vyvinout mnohem bohatší typy tříd, které budou omezeny pouze množinou vlastností jazyka.

Všechny jazyky jsou si rovny

Z pohledu .NET CLR jsou si všechny řízené jazyky rovny, neboť jsou založeny na stejné CLS (Common Language Specification) a bez výjimky na identickém CTS (Common Type System).

Zatímco některé jazyky poskytují jisté vlastnosti a jiné ne, jde vždy o vlastnosti jazyků, nikoli o vlastnosti CLS. Např. Visual Basic zpracovává názvy všech proměnných a funkcí jako symboly nezávislé na výšce písma.

Řízený kód má širokou podporu v průmyslu

Microsoft poskytuje tři jazyky generující řízený kód (Visual Basic.NET, C# a J# - implementace jazyka Java na platformě .NET), stejně jako rozšíření pro jazyk C++ generující řízený kód (Managed Extensions for C++).

Navíc několik dalších společností, vedle firmy Microsoft, vyrábí překladače generující řízený kód. Mezi další jazyky patří: Cobol, Haskell, Mercury, ML, Component Pascal, Perl, Python, Eiffel a Smalltalk.

Za účelem snížení pravděpodobnosti spuštění neřízeného kódu a eliminaci problémů spojených s nasazením aplikací dochází k úzké spolupráci mezi Systémy bezpečnosti a nasazení (Security and Deployment).

Jemně členěný bezpečnostní systém založený na faktech

.NET realizuje extrémně členitý bezpečnostní systém, umožňující řídit vše, co má být za běhu dovoleno konkrétním modulům kódu. V pozitivním případě dále určí, s jakými privilegii. Proces se týká i kódu staženého z Internetu.

Když je například .NET Framework aplikace stažena, žádá o jistou množinu povolení (jako je například zápis do dočasného adresáře). CLR shromáždí fakta (evidence) o aplikaci, jako je například místo stažení, jestli je aplikace podepsána Authenticode™ podpisem a dokonce, k čemu všemu chce přesně v systému přistupovat. Na základě této informace a ve spolupráci s administrativně nastavenými pravidly určí, jestli by mělo být povoleno spuštění aplikace nebo ne. CLR může dokonce říci aplikaci, že nemůže garantovat všechna požadovaná povolení a dát jí na výběr, jestli chce pokračovat v běhu nebo ne.

Souběžné nasazení

Díky existenci bezpečnostního systému založeného na faktech je velmi zjednodušena problematika nasazení aplikace. Jedním z největších problémů, kterému čelí vývojáři a administrátoři (a bezpochyby uživatelé) je správa verzí. Pokud vaše aplikace funguje jeden den a později ne z důvodu instalace nové, naprosto nezávislé aplikace, je velmi často důvodem přepsání některých sdílených knihoven dodaných společně s novou aplikací. Velmi často jde o opravy chyb, ale i změnu jistého chování, na které se existující aplikace spoléhají. Popsaný fenomén je znám pod označením DLL Hell.

.NET Framework obsahuje vylepšení, která prakticky odstraňují tento problém. Za prvé zahrnuje velmi silný interní jmenný systém poskytující odlišné jmenné prostory. Znamená to, že neexistuje žádný způsob, jak si splést dvě knihovny, i když nesou stejná fyzická jména souborů. Existuje také zcela nová vlastnost nazvaná „souběžné nasazení“ (side-by-side deployment). Pokud aplikace z předchozího příkladu skutečně přepíše sdílenou knihovnu, existující aplikace je schopna vlastní opravy. Ve chvíli, kdy je existující aplikace opakovaně spuštěna, kontroluje všechny sdílené soubory. Zjistí-li modifikaci v jednom z nich a tyto modifikace jsou nekompatibilní, požádá CLR, aby dodalo verzi o niž ví, že je kompatibilní. Díky systému bezpečnosti a řízení verzí CLR zvládne spolehlivě tuto operaci.

Bezpečnost je nedílnou součástí návrhu .NET

Verifikační kontroly začínají, jakmile je třída zavedena do paměti. Následovně jsou rozšířeny o kontrolu přístupu kódu ke zdrojům pomocí „přístupové bezpečnosti kódu“ (Code Access Security). Kontroly pronikají také do architektury CLR, aby zajistily dodržení hranic izolující aplikace.

Uvedené mechanismy pracují a rozšiřují bezpečnostní mechanismy běžně dostupné v soudobých operačních systémech. .NET Framework bezpečnost obecně zahrnuje následující oblasti:

- Typová bezpečnost: Typově bezpečné programy pracují pouze s pamětí, která byla alokována pro jejich použití a přistupuje k objektům pouze pomocí zveřejněných rozhraní. Z bezpečnostního pohledu umožňují odkazy na definovanou paměť bezpečné sdílení jednoho adresního prostoru několika objekty. Přístup k objektům pouze přes jejich veřejné rozhraní garantuje, že

bezpečnostní kontroly spojené s jistým rozhraním nelze obejít. CLR zajišťuje typovou bezpečnost kombinací silně typových metadat (parametry, členy, elementy polí, návratové hodnoty metod a statické hodnoty) se silnými typy v MSIL (lokální proměnné a sloty zásobníku). Tím jsou vytvořeny základy pro automatickou verifikaci typové bezpečnosti programů přeložených do MSIL, nezávisle na použitém zdrojovém jazyce. Taktéž v závislosti na místě, odkud dochází k zavedení kódu pomocí .NET CLR, je požadována verifikace před jeho spuštěním.

- Fakta (Evidence): Existují pouze dva způsoby, jak může být kód spuštěn. Za prvé pomocí Class Loader a zadruhé prostřednictvím služeb interoperability (viz dále v textu kapitoly). V obou případech jde o služby poskytované CLR, které jsou součástí „bezpečnostního opevnění“. Class Loader například udržuje informace o zdroji každé implementace, kterou zavede. Class Loader je pak schopen spolehlivě poskytnout jistá fakta, na nichž lze založit identitu kódu. Fakta mohou obsahovat informace jako je Internetová zóna a adresa, z níž kód pochází, jeho sdílené jméno a identitu vydavatele. S pomocí těchto informací může bezpečnostní politika řídit privilegia poskytnutá konkrétní aplikaci nebo sadě příbuzných aplikací.
- Bezpečnost přístupu kódu (Code Access Security): Tento mechanismus realizuje omezení založená na politikách definujících privilegia, která získají spuštěné assembly. Politiky mohou řídit přístup k systémovým zdrojům a jinému kódu na základě techniky povolení. .NET definuje celkem širokou množinu povolení vztahujících se k běžným systémovým zdrojům a k identitě kódu založené na faktech. Nástroje nebo vývojáři aplikací mohou tato povolení rozšířit a přidat nové typy povolení určené pro specifické potřeby. Povolení jsou logicky seskupena do množin povolení, která mohou být přiřazena bezpečnostním systémem .NET jednotlivým assembly. Takto přidělená povolení definují, co vše může kód vykonávat. Jsou určena na základě fakt spojených s kódem. Fakta jsou používána k určení sady skupin politik přiřazených dané assembly, která slouží bezpečnostnímu systému k určení poskytnuté množiny povolení. Povolení jsou typicky požadována důvěryhodným kódem (např. třídou), který zaobaluje přístup k lokálnímu souborovému systému. Většina aplikací většinou jednoduše zavolá tento řízený kód pracující s důvěryhodným zdrojem a nebude potřebovat žádný specifický kód pro řízení bezpečnosti.
- Povolení zdrojů (Resource Pemissions): Tato povolení jsou spojena s informacemi o komunikačních zdrojích uvnitř systému. Příkladem mohou být soubory, okna na obrazovce, schránka, síťové kanály, soukromá aplikační úložiště a schopnost volat neřízené API funkce. (Soukromé aplikační úložiště je realizováno třídou IsolatedStorageFile, novou množinou typů a metod podporovaných .NET platformou, specificky pro kód, který nemá přístup k souborovému systému.) Uvedené typy povolení mohou být použity k umožnění přístupu ke specifickému zdroji a to jak při zavádění, tak za běhu aplikace. V krátkosti si také proberme úzce spjatou deklarativní bezpečnost. Představme si aplikaci, která potřebuje číst a zapisovat do souboru v adresáři C:\Temp. Aby řízená aplikace otevřela soubor v tomto adresáři, použije základní třídu frameworku System.IO.File. Třída obsahuje statickou metodu nazvanou Open. Metoda Open interně získá cestu a jméno souboru, který chce uživatel otevřít a požádá jádro CLR o ověření přístupu ke konkrétnímu souboru. Tomu se říká vyžádání přístupu. Metoda Open požaduje, aby volající kód měl přiděleno FileIO povolení s přístupem pro čtení a zápis do C:\Temp*.* ještě před tím, než se pokusí otevřít soubor.

Ve chvíli, kdy metoda požaduje přístup, CLR projde zásobník volání a zjišťuje, jestli všechny assembly v řetězci volání mají požadované povolení. Jestliže některému kódu v řetězci volání chybí požadované povolení, pak žádost selže a je vyhozena bezpečnostní výjimka. V opačném případě je žádost úspěšná. Aby mohlo dojít k dokončení operace File.Open, třída File potřebuje zavolat API souborového podsystému poskytovaného operačním systémem. Volání vyžaduje nové povolení práce s neřízeným kódem, které je typicky přiděleno vysoce důvěryhodným třídám pracujícím se zdroji. U jiných assembly s částečnou důvěrou, která se nachází v řetězci volání, se nepočítá s tímto povolením. Aby došlo k úspěšnému uspokojení požadavku na práci s neřízeným kódem, u vysoce důvěryhodného kódu se předpokládá „přímého použití“ povolení. V jeho důsledku se prohledávání zásobníku volání ukončí v assembly, která povolení uplatňovala.

- Povolení identity (Identity Permissions): Tato povolení jsou založena na faktech spojených s danou assembly (doména, zóna, sdílené jméno a vydavatel). Umožňují vám řízení přístupu k metodám tříd na základě této informace. Povolení identity fungují stejným způsobem jako povolení zdrojů. Jestliže volající má požadované atributy identity, volání je úspěšné. V opačném případě je vyhozena bezpečnostní výjimka.
- Deklarativní bezpečnost: Tento výkonný mechanismus vám umožňuje vkládat bezpečnostní kontroly přístupu ke kódu přímo do aplikací formou anotace tříd, položek nebo metod. Deklarace jsou zakódovány v metadatech assembly a jsou zpracovány bezpečnostním systémem .NET. Aplikací deklarace můžete požádat o provedení kontrol buď při startu aplikace, nebo v průběhu jejího vykonávání. I přes své omezené možnosti jsou kontroly při startu aplikace mnohem úspornější, neboť k nim dochází pouze jedenkrát, narozdíl od kontrol prováděných při každém volání metody. Kontrolují pouze povolení (zdrojů nebo identity) bezprostředního volajícího a tak musíte pečlivě rozvážit, jestli jde o vhodný postup pro konkrétní situaci. Kontroly identity při startu aplikace jsou typicky určeny pro omezení přístupu ke třídám. Příkladem použití může být kontrola identity zajišťující, že pouze kód od určitého vydavatele nebo s jistým veřejným klíčem sdíleného jména může přímo přistupovat ke zvolené třídě. Použít těchto kontrol lze také k omezení možností zdědit nebo předefinovat virtuální metody.
  Deklarativní kontroly za běhu aplikace jsou jednoduché prostředky sloužící k indikaci požadavků a předpokladů povolení pro daný typ. Požadavky způsobují plný průchod zásobníkem pro určení existence povolení u každého kódu v řetězci volání. Jsou vhodné tehdy, když požadované akce a atributy povolení jsou neměnné. Pokud například vždy požadujete přístup pro čtení z adresáře C:\Temp, je jednoduché vyjádřit tento požadavek deklarativně. Avšak pokud potřebujete dynamicky měnit typ kontrol v závislosti na různých módech přístupu nebo jménech adresářů a souborů, pak budete potřebovat imperativní kontroly.
- Imperativní bezpečnost: Imperativní kontroly jsou jednoduše programově realizovány uvnitř metod. Jsou realizovány naprosto stejným způsobem jako deklarativní kontroly za běhu aplikace. Výhoda spočívá v podpoře dynamického určení konkrétních povolení, které jsou požadovány v daném exekučním kontextu. Důležité jsou zejména pro provolení jako FileIO a IsolatedStorage, které podporují více přístupových módů a dalších nastavení vztahujících se k používaným systémovým zdrojům.´
- Bezpečnost založená na politikách (Policy-based Security): Bezpečnost založená na politikách je nezbytným prvkem pro efektivní realizaci bezpečného přístup ke kódu. Politika vám umožňuje vytvářet specifická

prohlášení na téma, co by kódu mělo být dovoleno na základě místa jeho původu a dalších identifikačních informací. Při použití této politiky dochází k automatickému přidělení privilegií kódu, bez přímé interakce s uživatelem. Platforma .NET je dodávána s implicitními politikami, které odráží různé stupně důvěry typicky přidělované kódu na lokálním počítači, kódu ze známých důvěryhodných serverů a kódů z nedůvěryhodných internetových portálů. Politiky lze dodatečně uzpůsobit, pokud je potřeba.

- Bezpečnost založená na rolích (Role-based Security): Vývojáři velmi často provádějí bezpečnostní rozhodnutí na základě identity nebo rolí spojených s exekučním kontextem. Často vidíme tento případ u obchodních nebo finančních systémů, jako prostředek pro vynucení politik. Můžete například nastavit limity pro hodnoty transakcí v závislosti na roli, jakou sehrává uživatel vznášející požadavek. Úřednici mohou být schopni provádět transakce až do jisté výše, přímí nadřízení do vyššího limitu a manažeři do ještě vyšší hodnoty. .NET poskytuje služby, které umožňují aplikacím velmi jednoduše zahrnout uvedený typ logiky, a to způsobem nezávislým na typu platformy s podporou škálování v Internetu.
  .NET mechanismy jsou postaveny na principech identit (identity) a zmocnění (principal). Identity zaobalují pojmenování entit, kterým aplikace rozumí. Může tak dojít k jejímu mapování na interní účty operačního systému, ale mohou být také definovány aplikací. Odpovídající zmocnění zaobaluje identitu společně s odpovídajícími informacemi o rolích. Ta mohou být taktéž mapována na principu skupin v operačním systému.
  Během provádění operací důvěryhodný poskytovatel autentikace vypočte informace o zmocnění na základě pověření (credential), spojených s daným dotazem (HTTP Put, vzdálené volání procedury apod.). Zmocnění je následovně připojeno k exekučnímu kontextu a je přístupné aplikačnímu kódu, který provádí kontroly rolí nebo identit před provedením jistých akcí.
- Bezpečnost vzdáleného zpracování(Remoting Security): .NET poskytuje podporu pro vzdálený přístup k objektům nalézajících se v různých aplikačních doménách (AppDomain). Aplikační doména je logická aplikace a několik aplikačních domén může běžet v rámci jednoho Win32 procesu. Když dochází k překovávání hranic počítačů, stávají se bezpečnostní otázky jako autentikace, autorizace, utajení a integrita kritickými. .NET proto podporuje tyto mechanismy způsobem, který je kompatibilní s existujícími síťovými protokoly a velice úzce je integruje do infrastruktury vzdáleného volání. Díky nim je mnohem jednodušší přidat tyto vlastnosti do vašich aplikací.
  Autentikační mechanismy jsou vhodné pro identifikaci uživatelů, stejně dobře jako aplikací a obchodních entit. Identifikační infrastruktura založená na kryptografických klíčích slouží k řízení těchto identit a k integraci s autentikačními protokoly používající klíče. Služby pro správu důvěryhodnosti a integraci využívají kryptografické služby. Bezproblémově lze použít takové mechanismy jako jsou: Secure Socket Layer (SSL) a IP Security Protocol (IPSec). .NET Framework také podporuje aplikační vrstvy technologie šifrování, integrity a digitálních podpisů pro předávání zpráv.
- Kryptografie: .NET obsahuje množinu kryptografických objektů implementujících obecně známé algoritmy, které běžně používají hašování, kódování a generování digitálních podpisů. Objekty jsou navrženy způsobem, který podporuje začlenění základních schopností do mnohem složitějších operací, jako je podepisování a kódování dokumentů. Kryptografické objekty jsou používány interními službami .NET, ale jsou také dostupné vývojářům, kteří potřebují kryptografickou podporu.

Neviditelná infrastruktura

.NET Framework automatizuje při vytváření software infrastrukturní detaily na rozdíl od COM a umožňuje vám soustředit se na tvorbu aplikační logiky v libovolném jazyce.

Jedním z primárních cílů .NET Framework je zjednodušit vývoj komponent. Mezi nejtěžší aspekty komponentově orientovaného software na technologii COM patří práce s infrastrukturou COM. Aby se podařilo zjednodušit vývoj, .NET Framework automatizuje téměř všechny složité postupy tradičně spojené s COM vývojem, včetně počítaní referencí, popisu rozhraní a registrací.

.NET komponenty a COM komponenty jsou stejné

.NET komponenty lze volat z COM komponent napsaných ve Visual Studiu 6 a .NET komponenta bude vypadat jako běžná COM komponenta. I naopak, COM komponenta může být volána z .NET aplikací vyvinutých s Visual Studiem.NET a COM komponenty budou vypadat jako .NET komponenty.

Proti tomuto vztahu lze vznést námitku. Jako COM vývojář musíte udělat ručně mnoho věcí, na které se jako .NET vývojář může spolehnout a CLR je provede za vás automaticky. Například, pro plné zabezpečení COM komponenty musíte kód napsat manuálně a ručně řídit management paměti. COM komponentě také musíte nainstalovat odpovídající položky do Windows Registry. U .NET komponent CLR tyto vlastnosti automatizuje. Komponenty jsou samopopisné a můžete se instalovat bez registrace do Windows Registry.

.NET a COM+

Pojem COM+ se vztahuje k verzi COM, která kombinuje vlastnosti služeb Microsoft Transaction Serveru (MTS) s distribuovaným COM (DCOM). COM+ poskytuje množinu služeb orientovaných na střední vrstvu. Konkrétně COM+ poskytuje správu procesů, synchronizační služby, deklarativní transakční model a pooling objektů a databázových spojení.

COM+ služby jsou primárně orientovány na střední vrstvu vývoje aplikací a snahu poskytnout spolehlivé a ve velkém měřítku škálovatelné distribuované aplikace. Tyto služby jsou doplňkem k programovatelným službám poskytovaným .NET Framework a BCL (Base Class Library), k nimž zajišťuje přímý přístup.

Automatický management

.NET Framework posouvá možnosti COM+ a dosahuje mnohem vyšší produktivity vývojářské práce. .NET Framework například zavádí automatické řízení paměti, nezávisle na programovacím jazyce. Také vám poskytuje jeden unifikovaný programový model, který sbližuje dohromady nejlepší vlastnosti Windows® Fondation Classes, Microsoft Foundation Classes, Visual Basic API a doplňuje je o další možnosti, včetně pokročilé architektury pro práci s daty.

.NET Framework také využívá stejné infrastrukturní komponentové služby jako COM+, jakými jsou například transakce, queuing a pooling objektů. V budoucích verzích bude také obsahovat vlastnost nazvanou partitioning. Ta poskytuje silnou izolaci procesů a je určena pro poskytovatele aplikačních služeb. COM+ je nejvýkonnější a nejbohatší množinou aplikačních služeb dostupných v současnosti.

Využití existujících investic do COM

Microsoft vám umožňuje využít výhody existujících investic do technologie COM. Aplikace jsou schopny pracovat s novými schopnostmi .NET Framework a současně nasadit i existující COM komponenty.

Zatímco není nutné měnit existující kód, který bude nadále fungovat souběžně s řízeným kódem vyvinutým pomocí .NET jazyků, několik vlastností .NET Framework se výrazně změnilo. Vlastnosti, jako automatická správ paměti v .NET Framework a jeho schopnosti nasazení, jsou zcela odlišné od COM ekvivalentních počítání referencí a registračních mechanismů. Pro plné využití některých z uvedených vlastností bude nutné existující COM komponenty přizpůsobit. Může k tomu ale dojít až ve chvíli, kdy to bude výhodné na základě obchodních požadavků, plánu projektu a dostupnosti programátorů.

Spolupráce s neřízeným kódem

Prostředí řízeného kódu, poskytované CLR, obsahuje mnoho vlastností, které činí vývoj software potěšením. Operační systém, na kterém CLR běží však také nabízí své vlastní možnosti a bylo by velice nešťastné, kdyby řízeným aplikacím bylo zabráněno používat nativní služby operačního systému nebo neřízeného kódu.

Podpora tohoto požadavku je v CLR zprostředkována v jmenném prostoru System.Runtime.InteropServices. Jmenný prostor obsahuje množinu typů, umožňujících řízeným aplikacím přístup k neřízenému kódu. Jmenovitě se jedná o tři podporované scénáře spolupráce:

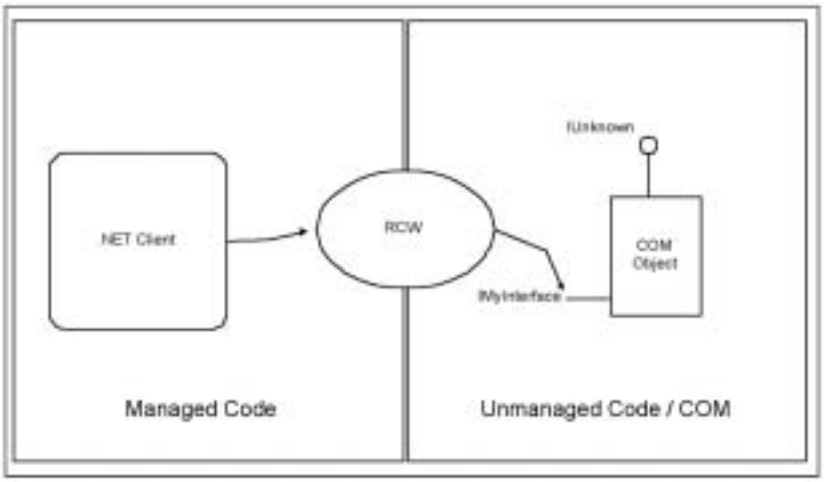
- Řízený kód volá funkce z neřízených DLL
- Řízený kód vytváří instance a volá metody rozhraní COM objektů a
- Neřízený kód vytváří instance a volá metody .NET objektů

Spolupráce .NET s funkcemi neřízených DLL

Pro vyvolání funkce neřízené DLL knihovny musíme oznámit CLR jméno funkce, která má být volána (například MessageBoxA), jméno DLL knihovny obsahující funkci (např. User32.dll) a jak mají být předány funkční parametry (např. datové typy parametrů mohou být vstupní, vstupní nebo vstup/výstupní).

Spolupráce .NET s COM

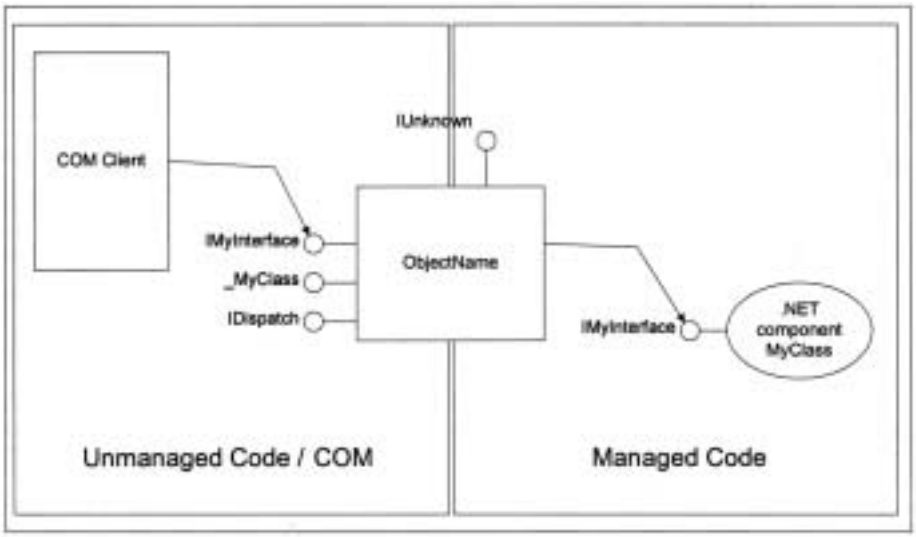
Při práci s COM objektem, musíme nejdříve vytvořit .NET obal (wrapper), aby se .NET klientům zdálo, že volají .NET kód. CLR bez problému zpracovává interní detaily. Utilita následovně analyzuje typovou COM knihovnu a vytváří řízenou DLL, jejíž metadata popisují metody zaobalující rozhraní COM objektu. Řízená aplikace může nyní založit instanci objektu a použít ji jako by byla nativním řízeným typem. Ke každému COM objektu CLR generuje Runtime-Callable Wrapper (RCW). Obal vystupuje jako prostředník (proxy) mezi řízeným a neřízeným kódem a řídí všechny administrativní úkoly, jako je předávání parametrů (marshalling), řízení životnosti atd. Obr. 15 ukazuje vytvořené RCW pro přístup ke COM objektu.



Obrázek 15 .NET Runtime Callable Wrapper

Spolupráce neřízeného kódu s .NET

K tomu, aby jste mohli používat řízené .NET objekty z neřízených aplikací, musíte přidat definici .NET třídy do Registry systému tak, jak COM prostředí předpokládá u lokálních COM objektů. .NET SDK podporuje nástroj pro registraci assembly (regasm), který generuje .NET objektům GUID a zapisuje informace do Registry. Vyžaduje-li neřízený kód typovou knihovnu, i tu je možné generovat. Nyní můžete vytvářet instance objektů v neřízených aplikacích a používat je jako by šlo o COM objekty. CLR generuje pro každý typ řízeného .NET objektu COM-callable Wrapper (CCW). Obal vystupuje jako prostředník mezi neřízeným kódem a řízeným kódem, stará se o administrativní úkoly jako je předávání parametrů (marshaling) a řízení životnosti. Obrázek 16 ukazuje CCW a interakci .NET komponent s COM klienty.



Obrázek 16 COM Callable Wrapper

Díky schopnostem spolupráce CLR se nemusíte vzdávat žádné z možností neřízeného kódu při současném využívání všech výhod, které CLR nabízí. CLR navíc zjednodušuje práci s COM objekty více než by tomu bylo uvnitř neřízeného C/C++ kódu.

Přechody mezi řízeným a neřízeným kódem mají vliv na výkonnost a tak by jste se je měli snažit omezit, kdykoli je to možné. Někdy to může znamenat převedení vašich COM objektů do .NET. Interoperabilita vám však umožní toto rozhodnutí učinit až ve chvíli, kdy se to bude hodit.

MOŽNOSTI NASAZENÍ

Platforma .NET využívá metadata a assembly k uložení informací o komponentách, umožňuje multijazykové programování a řeší neblaze proslulý problém DLL Hell. Metadata jsou použita pro jednoduché spojení a zavádění assembly.

Nasazení a řízení verzí s Assembly

.NET komponenty, které vytváříte, se nacházení v assembly a jsou uloženy v EXE nebo DLL souborech. Assembly se může skládat z jednoho nebo více fyzických souborů. Assembly je množina logicky příbuzných typů a reprezentuje jednotku pro řízení verzí a nasazení. Sloučením logické a fyzické podstaty opakovaně použitelných komponent můžeme rozdělit kód do několika souborů podle záměrů implementace. Lze například využít výhody postupného stahování na základě požadavku k nahrání pouze těch typů, které jsou nutné pro běh programu. Ve stejnou chvíli je stejná množina závislých typů udržována uvnitř assembly jako nedělitelná jednotka pro řízení verze a nasazení. Více souborové assembly se skládají z tzv. modulů. Modul je Portable Executable (PE) souborem (v formátu DLL nebo EXE), sestávající se z jednoho nebo více typů.

Assembly nám umožňují přesně vyjádřit závislosti mezi verzemi jednotlivých částí aplikace a díky tomu, že jsou samopopisné, napomáhají nasazení ve stylu „xcopy“.

Assembly také hrají významnou roli v bezpečnostním systému .NET, protože těmto jednotkám jsou přiřazena povolení a samy povolení také požadují. Při uplatnění politiky verzí, nasazení a bezpečnostních .NET charakteristik se assembly chovají jako jednotky definující rozsah pro vyřešení odkazů na typy.

Z pohledu uživatele assembly (externí pohled) je assembly pojmenovaná a očíslovaná verze kolekcí, exportující typy a zdroje. Z pohledu vývojáře (interní pohled) je assembly kolekcí jednoho nebo více souborů, které implementují typy a zdroje.

Metadata umožňují většinu rysů platformy

Metadata jsou lepidlem, které umožňuje mnohé z rysů Microsoft .NET Framework a jsou nesmírně bohatým popisem položek obsažených uvnitř assembly. Obsahují detaily jako jsou popisy typů a informace o libovolné externí assembly, kterou samotná assembly používá. Metadata také obsahují informace o verzi, popisy zdrojů (například obrázky ve formátu bitové mapy) uvnitř assembly a umožňuje realizaci dalších .NET charakteristik.

Metadata jsou používána CLR pro zjištění uložení a zavedení třídních typů ze souborů, rozložení instancí v paměti, vyřešení volání metod a odkazů na položky, překlad MSIL do nativního kódů, vynucení bezpečnosti a vykonání mnoha dalších operací.

Metadata třídy ji plně popisují, včetně jejich metod, parametrů metod, volacích konvencí, vlastností, událostí generovaných třídou a viditelnosti všech členů třídy. Metadata jsou také určena ke shrnutí všech atributů vystavených libovolným jazykem.

Metadata poskytují multijazykovou integraci
Jedním z největších půvabů .NET je výrazně zvýšená schopnost, s jakou lze jednoduše psát kód v různých jazycích. Aby toho bylo dosaženo, musí existovat společný formát pro zveřejňování informací popisující typy, sdílený všemi jazyky generující .NET kód. Metadata jsou právě tímto společným formátem.

Další výhoda použití metadat spočívá v tom, že jazyky na .NET platformě se mohly zbavit jazykově specifických mechanismů pro importování znalostí o externích komponentách. Rozličné jazyky se mohou lišit syntaxí importující metadata, ale nakonec všechny skončí u stejné informace.

Metadata umožňují vzájemnou spolupráci
Metadata jsou také rozhodující pro vzájemnou spolupráci .NET a Win32 API a COM objekty. Když řízený kód volá neřízené API, je použit mechanismus Plaftorm Invoke (PInvoke). .NET CLR musí vykonat speciální přípravy než provede PInvoke volání. Tento proces většinou představuje transformaci parametrů, například datové konverze, aby se zaručilo, že neřízený kód získá parametry v očekávaném formátu. Každá funkce a metoda volatelná z řízeného kódu musí mít asociována metadata.

Vedle poskytnutí jednotného způsobu importu data o externích komponentách, metadata také zprostředkovávají jednotný způsob jak popsat, na kterých assembly je komponenta závislá. .NET komponenty se na sebe neodvolávají pomocí Registry, ale formou metadat. Díky tomu si můžeme jednoduše dovolit, u většiny .NET aplikací, zkopírovat soubory do instalačního adresáře. Při deinstalaci aplikace soubory jednoduše smažeme.

Překladače generují metadata
Úkolem překladače je zpracovat zdrojový kód a vytvořit odpovídající MSIL. Vedle toho je překladač zodpovědný za umístění metadat do každé assembly.

Metadata jsou nadmnožinou starších technologií jako jsou Typové knihovny nebo Interface Definition Language (IDL). Za upozornění stojí fakt, že metadata jsou mnohem složitější než jejich předchůdci a jsou vždy spojena se souborem, který obsahuje kód. Jsou vždy připojena ke stejné assembly jako kód a stávají neoddělitelnými.

Po všech překladačích na .NET platformě je požadováno, aby generovaly plná metadata o každém typu přeloženém do zdrojového kódu modulu.

Nástroje jako Visual Studio.NET čtou informace metadat ze souborů pomocí technologie nazvané Reflection a realizují tak technologii IntelliSense™ uvnitř editorů zdrojových kódů.

Komponenty assembly jsou popsány v manifestu

Komponenty, z nichž je assembly sestavena, jsou popsány v manifestu. Manifest je blok dat, který realizuje výčet souborů assembly a řídí, jaké typy a zdroje jsou zveřejněny navenek assembly. Manifest také řídí, jak jsou reference na typy a zdroje mapovány na soubory obsahující deklarace a implementace. Dále provádí výčet ostatních assembly, na nichž je vlastní assembly závislá.

Existence manifestu realizuje jistou úroveň izolace mezi klientem assembly a implementačními detaily. Assembly se stávají samopopisné.

Assembly určují rozsah platnosti modulu

Každý modul zavedený za běhu aplikace je nahrán do kontextu assembly. Každý typ je zaveden do rozsahu platnosti určeného rozsahem assembly. Součástí identity typu je jeho přináležitost k assembly. Například, typ s názvem Klient, který je zaveden v rozsahu jedné assembly, není stejným typem, jako Klient zavedený v rozsahu jiné assembly, i přesto, že hash hodnoty jeho deklarace a implementace jsou identické.

Jako vývojář můžete v době návrhu provést explicitní nastavení. Importujete-li jistý typ z jiné assembly, metadata souboru vaší assembly budou obsahovat odkaz na tuto assembly. Pokud importujete typ přímo z jednotlivých modulů uvnitř multi-souborové assembly, vaše vlastní soubory obsahují explicitní odkazy definující závislost na těchto modulech. V druhém případě .NET ctí a chrání hranice assembly a zajišťuje, že váš kód a uvedené moduly jsou součástí stejné assembly.

Assembly Cache zvyšuje výkonnost

Assembly cache je systémový adresář sloužící pro ukládání assembly, které mohou být použity více než jednou aplikací. Když jsou sdílené assembly na počítač instalovány, mohou být umístěny do assembly cache. Assembly cache obsahuje dvě logicky oddělené cache: Global Assembly Cache (GAC) a krátkodobou assembly cache.

Při distribuci assembly může dojít k její instalaci do GAC. Jestliže jsou assembly stahovány pomocí Internet prohlížeče, assembly je instalována do krátkodobé assembly cache. Vytvořením logického oddělení assembly cache zabrání nepříznivému ovlivnění instalovaných aplikací staženými moduly.

VÝHODY PRO PROGRAMÁTORY

Microsoft .NET se snaží vybavit programátory novou vývojovou platformou proti níž lze programovat a nahradit tak přímé programování nad operačním systémem. Nejznatelněji výhodu .NET přístupu ucítí vývojáři.

Programováni Internetu pomocí webových XML služeb

Ve chvíli, kdy vytváříte distribuované webové aplikace na .NET, programujete je pomocí webových XML služeb, usnadňujících oslovení výzvy integrace aplikací. Pomohou vám například vzít odlišné aplikace, běžící na různých operačních systémech, s odlišnými objektovými modely, napsané v různých programovacích jazycích a převést je na webové aplikace s jednoduchým použitím.

Obdobně jako komponenty, reprezentují webové služby funkcionalitu černé skřínky, která může být opakovaně použita bez obav týkajících se interní implementace. Aplikace lze sestavit kombinací vzdálených služeb, lokálních služeb a zákaznicky specifického kódu.

Sestavit online obchod můžete například pomocí služby Microsoft Passport pro autentikaci uživatelů, personalizační služby třetí strany pro přizpůsobení web stránek individuálním uživatelům, služby pro práci s kreditními kartami, daňové služby, služby pro sledování balíků realizovanou dodávkovou firmou, firemní katalogové služby připojující interní aplikaci firemní skladu a části zákaznického kódu, který ochrání váš obchod před davy uživatelů.

Na rozdíl od současných komponentových technologií, nepoužívají webové služby specifické protokoly pro objektový model. Místo toho komunikují pomocí všudypřítomných webových protokolů a datových formátů jako jsou HTTP a XML. Jakýkoli systém podporující webové standardy je schopen podporovat webové služby.

Navíc, kontrakt webové služby, který je součástí webové služby, popisuje poskytovanou službu ve smyslu zpráv, které webová služba akceptuje a generuje. Výhradním zaměřením na otázku zpráv se stává model webových služeb zcela jazykově, platformově a na objektovém modelu nezávislým.

Datový přístup pro webové služby

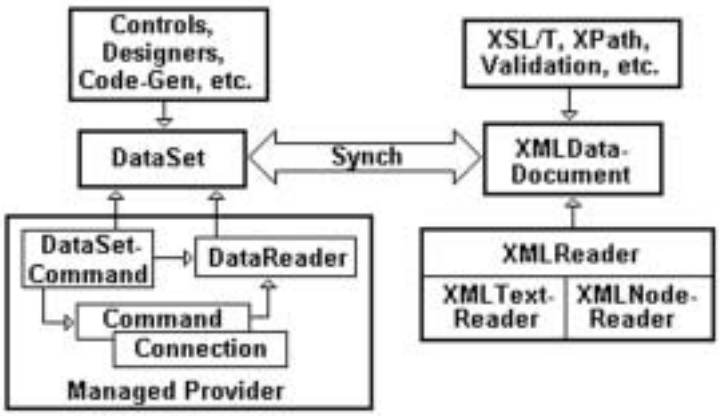
Téměř všechny webové služby potřebují číst nebo občerstvovat uložená data, a to buď v jednoduchých soborech, relačních databázích nebo jiných datových skladech. Přístup k datům je ve službách .NET Framework realizován knihovou tříd ADO.NET. Jak název napovídá, ADO.NET se vyvinulo z ActiveX™ Data Objects (ADO).

ADO.NET je navrženo za účelem poskytování služeb přístupu k datům pro škálovatelné webové aplikace a služby. ADO.NET nabízí vysoce výkonné proudové API pro připojený, kurzorově orientovaný datový přístup, stejně jako odpojený model daleko vhodnější pro předávání dat klientským aplikacím. V architektuře ADO.NET mohou být data (nezávisle na tom, jak jsou ve skutečnosti uložena) manipulována jako XML nebo jako relační data, podle toho, co je v danou chvíli pro aplikaci nejvýhodnější.

ADO.NET definuje třídy pro připojení k databázím, zadávání dotazů a čtení výsledků z datových skladů. Tyto třídy jsou implementovány pomocí řízených poskytovatelů dat (Managed Data Providers). ADO.NET objekty Connection a Command vypadají velice podobně jako jejich ekvivalenty v ADO a nová třída nazvaná DataReader realizuje schopnost pořídit výsledky pomocí vysoce výkonného proudového API. DataReader je funkcionální ekvivalent Forward-only, Read-only ADO RecordSetu, avšak

DataReadery jsou navrženy s minimalizací počtu objektů v paměti, pro minimalizaci nároků na garbage collection a zvýšení výkonnosti. .NET Framework obsahuje řízené poskytovatele dat pro Microsoft SQL server a libovolné datové sklady dostupné přes OLE DB.

Jednou z největších inovací ADO.NET je zavedení Datasetu. Dataset je paměťovou datovou cache, poskytující relační pohled na data. Sestává se z jedné nebo více tabulek, společně s připojenou tabulkou relací a omezení. Datasety neví nic o zdrojích jejich dat. Data mohou být založena nebo naplněna programově nebo nahráním z datového skladu. Nezávisle na tom, odkud data pocházejí, je s nimi zacházeno stejným programovým modelem a s využitím stejného interního principu cache. Obrázek 17 ilustruje základní architekturu ADO.NET.



Obrázek 17 Architektura ADO.NET

Pokročilý vývoj webových aplikací pomocí ASP.NET

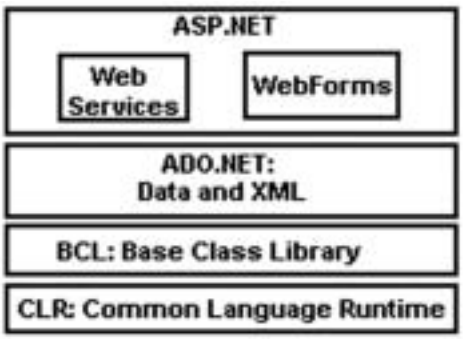
ASP.NET je posledním vylepšením současné technologie Active Server Pages (ASP). ASP.NET však bylo vytvořeno zcela od počátku, a tak získáváte výhodu zcela nových a významně vylepšených vlastností, jakými jsou:

- Objektově orientované kompilační jazyky
- Čisté oddělení kódu od obsahu
- Vysoká výkonnost a škálovatelnost
- Web Forms pro rychlý vývoj webových stránek, včetně událostního modelu a webových ovládacích prvků
- Různé úrovně podpory cache
- Serverové ovládací prvky, které umí generovat různé výstupy, jako je HTML 3.2, v závislosti na typu klienta. Pokročilí klienti, včetně Internet Explorer, mohou vyžít širší funkcionality
- Zjednodušený vývoj a přístup k webovým službám
- Přístup k bohaté platformě pomocí .NET Framework

ASP.NET stránky mají velké množství nových extenzí. Základní ASP.NET stránky jsou označeny příponou .aspx. Webové služby používají .asmx a nové konstrukce nazvané uživatelské ovládací prvky ascx. Nové souborové extenze jsou zavedeny z důvodu umožnění paralelního běhu stávajících ASP aplikací vedle ASP.NET aplikací. Oba typy aplikací spolu nesdílí stav připojení nebo aplikační stav a v míru existují na jednom serveru. Nové souborové přípony jsou vyžadovány Internet Information Serverem (IIS) pro vyvolání odpovídajících ISAPI filtrů pro jejich zpracování.

Další nové vlastnosti obsahují dodatečné serverové komentáře (<%- -%>, které umožňují zakomentovat celý blok kódu, výrazy připojení k datům (<%# %>) realizující spojení serverových ovládacích prvků s daty a nové direktivy (%@ %) umožňující deklarativní nastavení řady parametrů stránky.

ASP.NET využívá výhody CLR a služeb frameworku pro realizaci spolehlivého, robustního a škálovatelného hostujícího prostředí webových aplikací. ASP.NET také čerpá z výhod CLR modelu assembly a zjednodušuje nasazení aplikací. Navíc poskytuje služby pro jednoduchý vývoj aplikací (jako jsou služby řízení stavu) a vysokoúrovňový programový model (např. ASP.NET Webové formuláře a ASP.NET Webové služby). Obrázek 18 ukazuje systémovou architekturu ASP.NET.



Obrázek 18 Architektura ASP.NET

Čistější kód

ASP.NET nabízí vlastnost nazvanou „kód v pozadí“ (code behind), který vám umožní separaci specifického prezentačního obsahu od aplikační logiky. Kód metod událostí lze uložit do samostatného souboru, odlišného od .aspx souboru s prezentačním kódem (typicky HTML). Logiku vaší aplikace lze vyvinout pomocí libovolného .NET jazyka, běžícího na CLR. Znamená to, že kód ASP.NET má přístup ke všem jeho výhodám (například ladění, překlad, dědičnost, bezpečnost, garbage collection, škálovatelnost, jednoduchost nasazení atd.). Rozdělení kódu do samostatných souborů jej výrazně zčitelňuje. Další výhodou je, že oddělení grafického návrhu může dát portálu profesionální vzhled, zatímco programátoři se mohou postarat o obchodní logiku.

Existují i další výhody, vedle oddělení obsahu od logiky. Jedna z nich spočívá v možnosti použití standardních design nástrojů při konstrukci uživatelského rozhraní, bez nutnosti přidávat složitý kód. Další spočívá v jednoduchosti lokalizace díky tomu, že stránky s uživatelským rozhraním obsahují čistý HTML, místo promíchaného HTML a kódu.

Veškerý ASP.NET kód je nyní překládán a není již za běhu interpretován. Ačkoli jsou ASP.NET stránky plně překládány, lze ASP.NET stránky bez problému zaměnit, aniž bychom restartovali IIS proces. Jednoduše si lze udržet snadný ASP styl vytváření a nasazení stránek, při využití plně kompilovaného kódu. Posun k pevně typovým jazykům také znamená, že již nejste nadále omezeni datovým typem variant. Nyní lze deklarovat silně typové proměnné a objekty a zajistit brzkou vazby mezi nimi.

Výrazné zvýšení výkonu ucítíte, když přepíšete ASP kód a změníte všechny proměnné z volně typového variantu na silně typové datové typy. Tato změna sama o sobě může způsobit výrazné zvýšení výkonnosti. Navíc, přepsáním existujících COM

komponent na řízený kód odstraní mnoho výkonnostní provize, spojené s předáváním parametrů, řízením threadů a spoluprací s COM. ASP.NET také zavádí několik zásadních změn v sémantice stránky. Z pohledu kompatibility a migrace jsou nejdůležitější následující:

- Na stránce lze použít pouze jeden jazyk. V rámci jedné stránky však můžeme beze změny používat klientské skripty v jazyce JavaScript a serverový skript ve Visual Basic.NET. Všimněte si, že uživatelské ovládací prvky (nová vlastnost, umožňující zakládat opakovaně použitelné elementy, které mohou být vloženy do jiných stránek) mohou být napsány v jiném jazyce než je hostující stránka.
- Funkce musí být uzavřeny ve skript blocích (pokud používáte vložený kód místo souborů s kódem v pozadí). Například místo uvedení:

```
<%
        Function MojeFunkce()
        ......
        End Function
%>
```

Musíte nyní napsat:

```
<SCRIPT LANGUAGE="VB" RUNAT="SERVER" >
        Function MojeFunkce()
        ......
        End Function
</SCRIPT>
```

- Funkce výpisu nejsou podporovány. Na základě implementace ASP bylo možné napsat následující funkci, která využívá bočního efektu skriptování.

```
<% Function NakresliTabulku () %>
  <table>
        <tr>
              <td>
              <% Response.Write Nazdar Svìte %>
              </td>
        </tr>
  </table>
<% End Function %>
```

V ASP.NET musí být všechny funkce uzavřeny ve script blocích, a tak neexistuje žádný způsob, jak jednoduše vložit HTML. Výše uvedený kód musí být změněn následujícím způsobem:

```
< SCRIPT language="vb" runat="server" >
  Function NakresliTabulku()
        Response.Write <table><tr><td>
        Response.Write Nazdar Svìte
        Response.Write </td></tr></table>
  End Function
</SCRIPT>
```

Kompatibilita prohlížeče

Kompatibility s prohlížečem je výhodně dosaženo díky technologii ASP.NET, využívající webové formuláře se serverovými ovládacími prvky, které automaticky generují HTML kód. V programovém modelu webového formuláře navrhnete uživatelské rozhraní pomocí serverových ovládacích prvků, které se spouštějí na straně serveru. Ovládací prvky jsou schopny za běhu detekovat typ prohlížeče a poslat zpět HTML verze 3.2 nebo dynamický HTML, jak je potřeba. Důležitou skutečností zůstává, že logiku vyjádříte pouze jedenkrát formou událostí vyvolaných na straně serveru.

V jazyce Visual Basic umístíte na formulář ovládací prvek a implementujete proceduru události. ASP.NET poskytuje stejný programový model. Místo nutnosti manuálního čtení hodnot z proměnných formuláře, můžete napsat následující kód:

```
<html>
   < script language="vb" runat="server" >
     Sub TlacitkoOdeslat_Stisk (Source As Object,
       E As EventArgs)
       VasText.Text = Zadal jste & Jmeno.Text
     End Sub
   </script>
   <body>
   <form method="post" runat="server">
     Jméno: <asp:textbox id="Jmeno" runat="server" /><br>
     <asp:Button text="Odeslat"
       OnClick="TlacitkoOdeslat_Stisk"
       runat="server" />
     <br>
     <asp:label id="VasText" runat="server" />
   </form>
   </body>
</html>
```

Všimněte si použití atributu runat=„server" a elementů s prefixem jmenného prostoru asp:. Webové ovládací prvky jsou identifikovány formátem asp:element a atribut runat=„server“ naznačuje, že jde o serverové ovládací prvky, zpracovávané na straně serveru.

Všimněte si také události OnClick. Událost je vyvolána na straně serveru díky atributu runat=„server“ přidaného k odpovídajícímu ovládacímu prvku. Pozorujte, jak se atribut OnClick, nastavený u prvku asp:Button, spojí s funkcí zpracující událost. V našem případě TlacitkoOdeslat_Stisk.

Také si povšimněte, jak se můžete odvolávat na ovládací prvky uvnitř kódu vykonávaného na serveru. Například, kód funkce zpracování události nastavuje vlastnost Text ovládacímu prvku asp:label pomocí jeho ID.

Kód mnohem více připomíná tradiční formulářovou aplikaci v jazyce Visual Basic a je mnohem intuitivněji vytvořen než kód u ekvivalentní ASP logiky. Pro připojení procedur událostí k serverovém kódu používáme serverové ovládací prvky. Tyto prvky zasílají čistý HTML nízko-úrovňovým prohlížečům bez jakéhokoli klientského skriptu.

Ačkoli je implicitním výstupem ASP.NET webových ovládacích prvků kód HTML 3.2, mohou také pro odpovídající prohlížeče generovat DHTML. Vzniká tak možnost navrhovat stránky pomocí jednotné množiny serverových ovládacích prvků a nechat prvky samotné rozhodnout, co budou posílat na výstup v závislosti na typu prohlížeče. Umožní se tím zasílání DHTML a klientských skriptů novým verzím Internet Exploreru a kódu HTML verze 3.2 ostatním prohlížečům.

Nestavový model

Web funguje ze své podstaty v nestavovém modelu, bez jakékoliv závislosti mezi jednotlivým HTTP dotazy. Vytváření webových aplikací může být složitější, neboť aplikace si někdy potřebují uchovat stav mezi několika dotazy.

ASP.NET vylepšuje služby řízení stavu zavedené v ASP a poskytuje tři různé typy stavu ve web aplikacích: aplikační (application), relace (session) a pohledu (view).

- **Application state.** Je stejný jako v ASP, specifický pro danou instanci aplikace a nedochází k jeho perzistenci.
- **Session state.** Je specifický pro relaci mezi aplikací a uživatelem. Na rozdíl od stavu relace v ASP, může být stav v ASP.NET uchován v samostatném procesu nebo dokonce konfigurován pro uložení na samostatném počítači. Umožní se tím možnost využití stavu relace i v případech, kdy je aplikace nasazena na webové farmě.
- **View state.** Vztahuje se ke stavu stránek a jejich aktuálnímu nastavení, společně se stavem libovolného ovládacího prvku umístěného na stránce. Připomíná stav relace, avšak nedochází k jeho expiraci a je automaticky předáván v datech při každé komunikaci mezi serverem a klientem. Stav pohledu je také výhodný pro ukládání uživatelských nastavení a dalších osobních informací.

Stav pohledu si demonstrujme na ukázce HTML 3.2 kódu vygenerovaného z předchozího příkladu.

```
<html>
<body>
<FORM name="HtmlForm2 method="post" action="Test.aspx" id="HtmlForm2"
>
   < INPUT type="hidden" name="__VIEWSTATE" value="a0z664351470__x" >
   Jméno: < input name="Jmeno" type="text" id="Jmeno" >
   <br>
   <input type="submit" name="Button5" value="Odeslat">
   <br>
   <span id="VasText"></span>
</FORM>
</body>
</html>
```

Kód realizuje standardní formulář, který zasílá uživatelský vstup zpět do stejného souboru. Žádný stav není uchováván na serveru a neexistuje ani klientský kód, který by jej řídil. Stav ovládacích prvků je udržován ve skryté položce __VIEWSTATE. Znamená to, že ovládací prvky automaticky uchovávají svůj stav mezi jednotlivými post-back operacemi, bez jakékoliv programátorského vysílení.

ASP.NET podporuje nejen přesunutí řízení stavu do odděleného procesu, ale i na jiný počítač. Znamená to, že můžete vytvořit speciální server, zpracovávající stav pro několik serverů spojených do webové farmy. I přes existenci výkonnostního poklesu při přesunutí stavu relace do samostatného procesu a významné výkonnostní režie při přesunutí na jiný počítač, jsou výhody obrovské. Stav relace může nyní přežít restart aplikace, stejně jako restart počítače. Může dokonce fungovat přes hranice firewallu a po směrovaných spojeních.

Byl vytvořen i druhý typ poskytovatele stavu relace, který využívá SQL server jako trvalé úložiště dat. Ačkoli zaplatíme ještě vyšší výkonnostní cenu, varianta umožňuje datům relace přežít restart samotného stavového serveru.

Jedním z největších omezení stavu relace byla jeho závislost na cookies, která připojovala uživatele k jejich uloženému stavu na serveru. ASP.NET nyní nabízí stavový model bez cookie (cookie-less model), který funguje tak, že přidává informace, běžně přenášené v cookie, do URL.

Webové formuláře

Webové formuláře přináší výhody produktivity známé z Visual Basic formulářů do oblasti vývoje webových aplikací. Webové formuláře umožňují rapidní vývoj programovatelných webových aplikací nezávislých na typu platformy nebo prohlížeče, pomocí stejných technik, jaké známe z předchozího vývoje formulářových dektopových aplikací.

Webové formuláře podporují tradiční ASP syntaxi míchající HTML obsah se skript kódem, ale zdůrazňují mnohem strukturovanější přístup, který odděluje aplikační kód od obsahu uživatelského rozhraní. Standardní webové formuláře se skládají z HTML souboru obsahujícího visuální reprezentaci stránky a zdrojový soubor s kódem pro zpracování události. Zdrojový kód je přeložen do spustitelného kódu zvyšující výkon za běhu aplikace. Oba soubory zůstávají na serveru, kde jsou i spouštěny a generují HTML 3.2 kompatibilní dokument zasílaný klientovi.

Konstrukce windows formuláře ve Visual Basicu znamená přidání formuláře do projektu, přetažení ovládacího prvku na jeho plochu a následujícím dvojklikem na prvek vytvoření kódu v pozadí formuláře. Webové formuláře poskytují stejné postupy návrhu a úrovně produktivity do prostředí Webu.

Ovládací prvky webový formulářů zakrývají složitost udržení stavu při interakci uživatele se stránkou. Existují i bohaté služby připojení k datům, které vývojářům výrazně zjednodušují proces zobrazení dat získaných pomocí služeb přístup k datům.

Vyšší výkonnosti dosáhneme na ASP.NET stránkám oddělením kódu od obsahu a jejich dynamickým překladem do řízeného kódu. Každý HTTP dotaz je doručen nové instanci stránky, a tak se vývojáři nemusí zabývat threadovou bezpečnosti uvnitř svého kódu.

Webové formuláře z pohledu návrhu:

- Využívají programový model, který vývojáři již dlouho znají. Vývojáři mohou také velice jednoduše přecházet mezi webovým a desktopovými projekty bez potřeby intenzivního tréninku.
- Oddělují HTML vzhled od kódu v pozadí na stránce. Tato separace zjednodušuje nezávislé provedení změn v každém ze souborů, simplifikuje kód a umožňuje snadnější vytváření nových verzí kódu.
- Mohou být navrženy pomocí libovolného nástroje ve Visual Studiu.NET. HTML stránky mohou být také importovány nástroji jako FrontPage a převedeny na webové formuláře.

Webové formuláře mají i výhody za běhu:

- Poskytují programový model a exekuční prostředí, které umožňuje vývojářům implementovat HTML stránky generované na straně serveru.
- Výrazně zvyšují výkonnost, protože stránky obsahující kód v pozadí nejsou interpretovány, nýbrž kompilovány.
- Pro starší typy prohlížečů generují HTML stránky, které jsou čistým HTML 3.2. Ve výsledném efektu můžeme na stránky přistoupit z libovolné platformy.

Serverové ovládací prvky

ASP.NET zaobaluje velké množství funkcionality formou serverových prvků. Například došlo k nahrazení ovládacího prvku pro rotaci reklamních proužků (ad rotator), který nyní používá XML pro uložení informací. Ovládací prvky jsou také použity k řízní stavu na formulářích, zobrazení kalendáře a tabulek. Téměř ke každému HTML elementu existuje ASP.NET ovládací prvek, se kterým lze programově pracovat. Nemusíte například nadále programovat kód, obsahující smyčku s podmínkou, která udržuje vybranou položku list boxu. Jednoduše lze list boxu říci, aby běžel na serveru a vše je automaticky zařízeno za vás. Programově také můžete list boxu říci, která ze zobrazených položek bude vybrána.

Pravděpodobně jedním z nejzajímavějších nových ovládacích prvků je DataGrid. Jde o vícesloupcový grid s datovým zdrojem, který lze velice snadno připojit k ASP.NET datasetu. Podporuje stránkování, třídění a individuální editaci jednotlivých sloupců za použití velice malého množství kódu.

Serverové ovládací prvky jsou novinkou, ale není složité se je naučit, neboť jsou mapovány na jejich HTML protějšky. Serverové ovládací prvky se poznají podle jmenného prostoru asp: a používají XML přístup končícího lomítka při uzavírání elementu. I když nejste omezeni na XML syntaxi a můžete používat HTML styl se samotným uzavíracím tagem (např. </asp:TextBox>), XML je mnohem rychlejší. Jmenný prostor musí být použit protože definuje, odkud ovládací prvky pocházejí. Všechny standardní webové ovládací prvky jsou součástí jmenného prostoru asp. Je to důležité ve chvíli, kdy vytváříte své vlastní ovládací prvky.

Může se zdát, že ovládací prvek TextBox nenabízí příliš mnoho výhod před standardním input boxem, ale zkuste zvážit následující tři ovládací prvky vstupu:

```
<input type="text" ...>
<input type="password" ...>
<textarea rows="5" ...>
```

Všechny jsou v HTML používány pro realizaci vstupu, ale neexistuje zde žádná jednotnost. Posuďte, jak jsou následující řádky mnohem jednodušší:

```
<asp:TextBox runat="server" ...>
<asp:TextBox mode="Password" runat="server" ...>
<asp:TextBox Rows="5" runat="server" ...>
```

Jeden jednoduchý ovládací prvek může nahradit funkcionalitu třech, v závislosti na tom, jak je použit. Ve skutečnosti ovládací prvky typu TextBox generují stejný HTML výstup. Je však mnohem jednodušší si je zapamatovat a pracovat s nimi.

ASP.NET je dodáván s pěti rodinami serverových ovládacích prvků:

- skutečné ovládací prvky, které jsou mapovány na jejich HTML ekvivalenty,
- ovládací prvky seznamů, umožňující zobrazení dat na stránce,
- pokročilé ovládací prvky, poskytující komplexní uživatelské rozhraní a funkcionalitu,
- validační ovládací prvky pro vykonávání různých typů kontrol,
- mobilní ovládací prvky, zaobalující WML a WAP pro mobilní zařízení.

Skutečné serverové ovládací prvky jsou podobné HTML prvkům, došlo však u nich k odstranění nelogického chování a dosažení jednotného použití. Tyto ovládací prvky zahrnují LinkButton, ImageButton, HyperLink, TextBox, CheckBox, RadioButton, DropDownList, ListBox, Image, Label, Panel, Table, TableRow a TableCell.

Ovládací prvky seznamů obsahují Repeater, DataList a DataGrid. Množina také obsahuje prvky RadioButtonList a CheckBoxList pro snadnější vytvoření seznamu radiových tlačítek nebo zaškrtávacích tlačítek.

Pokročilé ovládací prvky zahrnují Calendar a AdRotator. Ovládací prvek Calendar zobrazí čistý HTML na jednoduchých prohlížečích nebo DHTML na pokročilých prohlížečích. AdRotator zobrazuje obrázky a má zabudovaný kód pro jejich rotaci.

Validační prvky zahrnují ReqiredFieldValidator, CompareValidator, RangeValidator, RegularExpressionValidator, CustomValidator a ValidationSummary. Prvky realizují jednoduchý způsob, jakým mohou vývojáři realizovat kontrolu uvnitř formulářů.

Je důležité si uvědomit, že serverové ovládací prvky nejsou COM ovládacími prvky nebo design-time prvky. Jsou konstruovány tak, že existují pouze na straně serveru a dovedou se vykreslit na klientovi jako standardní HTML.

Objektově orientovaný model

ASP.NET byl přepsán s myšlenkou objektové orientace. ASP.NET poskytuje objektový model stránek. Na vrcholu objektové hierarchie se nachází objekt Page. Znamená to, že každá stránka je instancí objektu Page. Událost Load třídy Page je jednou z mnoha důležitých událostí, na které reagujete:

```
<html>
<script language="vb" runat="server" >
   Sub Page_Load (Source As Object, E As EventArgs)
      ' kód provádìný pøi nahrání stránky
   End Sub

   Sub TlacitkoOdeslat_Stisk (Source As Object, E As EventArgs)
      ' kód provádìný pøi stisku na tlaèítko Odeslat
   End Sub

   Sub Page_Unload (Source As Object, E As EventArgs)
      ' kód provádìný pøed zrušením stránky zpamìti
   End Sub
</script>

<form runat="server" >
   <asp:Button text="Odeslat" OnClick="TlacitkoOdeslat_Stisk"
     runat="server" />
   <asp:Label id="VasText" />
</form>
</html>
```

Událost Load je vyvolána, jakmile dojde k nahrávání stránky a všechny serverové ovládací prvky jsou připraveny k použití. Během interakce s uživatelem jsou generovány ostatní události a na závěr, ve chvíli rušení stránky, je vyvolána událost Unload.

Každý objekt na stránce může mít svůj vlastní událostní model, zveřejnit jej a vyvolávat serverové události zpracované vašimi vlastními metodami na serverové straně. Procedury jako jsou Button_Click nebo ListBox_Change mohou relativně usnadnit zpracování běžných operací na standardních formulářích. Dochází také k významnému zčitelnění kódu.

Uživatelské ovládací prvky
V žádném případě nejste omezeni dodávanými ovládacími prvky a je velice snadné si vyrobit své vlastní. Pokud například chcete zaobalit do jednoho prvku dva text boxy, například pro zadání položek jména a příjmení, mohli by jste napsat podobný kód:

```
<asp:Panel runat="server" >
   <asp:TextBox id="txtJmeno" text="Jméno"
     runat="server" />
   <asp:TextBox id="txtPrijmeni" text="Pøíjmení"
     runat="server" />
</asp:Panel>
```

Kód můžete uložit do souboru pod názvem Jmeno.ascx (všimněte si nového typu přípony) a chápat jej jako by to byl webový ovládací prvek. Později můžete například přidat následující webový formulář:

```
<%@ Register TagName="Jmena" TagPrefix="ABC" Src="Jmeno.ascx" %>
<form>
   <ABC:Jmena runat="server" />
</form>
```

Velice jednoduše můžete vytvořit opakovaně použitelný kód. Ovládací prvky můžete také přímo naprogramovat v některém z .NET jazyků, odvodit je z jiných prvků a nechat je kreslit na výstup cokoli potřebují. Protože ovládací prvky jsou umístěny do jmenných prostorů, neměly by nastávat kolize mezi jejich jmény. Programové prostředí se tak stává velice dobře rozšiřitelné. Je vysoce pravděpodobné, že se objeví velké množství komplexních prvků třetích stran.

Jiným způsobem oddělení kódu od stránky je jeho umístění do knihovny a následná modifikace bloku script tak, aby se na knihovnu odkazoval. Ukázka následuje:

```
<script runat="server" src="MujKod.dll" >
```

Takový výraz dynamicky a automaticky připojí všechny události ke kódu přeloženém do MujKod.dll.

Ovládací prvky pro práci s daty

Jedním z nových webových ovládacích prvků je DataGrid, který má zabudovánu podporu pro vykreslení množiny dat. Pro vygenerování HTML tabulky na základě SQL dat, jednoduše založíme ADO.NET objekty a spustíme odpovídající příkaz pro získání dat. Ty jsou použity jako datový zdroj pro objekt data gridu.

```
<%@ Import Namespace="System.Data.SqlClient" %>
<html>
<script language="vb" runat="server" >
   Sub Page_Load (Sender As Object, E As EventArgs)
     Dim myCommand As SqlCommand
     myCommnad = New SqlCommand (select * from products;
        Server=localhost; Database=AdvWorks; UID=sa )
     DataGrid.DataSource = myCommand.Execute
     DataGrid.DataBind
   End Sub
</script>

<body>
   <asp:DataGrid id="DataGrid1" runat="server" />
</body>
</html>
```

Vše, co musíte udělat, je připojit zdroj dat k data gridu a HTML tabulka je vytvořena. Připojení k datům není omezeno pouze na data z databází. Můžete připojit hash tabulky, pole, jiné ovládací prvky, vlastnosti na stránce a XML.

Pokud vám nevyhovuje implicitní množina sloupců, můžete si nastavit zobrazení pouze těch, o které máte zájem.

```
<asp:DataGrid id="DataGrid1" AutoGenerateColumns="false"
runat="server" >
   <Columns>
     <asp:BoundColumn HeaderText="Jméno" DataField="ProductName" />
     <asp:BoundColumn HeaderText="Popis"
DataField="ProductDescription" />
   </Columns>
</asp:DataGrid>
```

Ovládací prvky Repeater a DataList využívají tzv. šablony (templates) umožňující přizpůsobovat jejich vzhled. Jak Repeater, tak DataList nemají implicitní uživatelské rozhraní, a tak jim musíme poskytnout šablonu. Šablony jsou pro oba typy prvků zapsány velice podobným způsobem:

```
<asp:DataList id="DataList1" runat="server" >
   <HeaderTemplate>
     Tady je seznam titulù <br>
   <HeaderTamplate>
   <ItemTemplate>
     <%# DataBinder.Eval (Container.DataItem, Title) %> <br>
   </ItemTemplate>
</asp:DataList>
```

Pomocí takové šablony lze specifikovat, které HTML ovládací prvky budou tvořit jednotlivé části vaší datové komponenty. Existuje pět různých jmen šablon, které můžete použít u prvku DataList: HeaderTemplate na horní část prvku, ItemTemplate pro každou položku, AlternatingItemTemplate pro každou druhou položku, SeparatorTemplate pro plochu mezi jednotlivými položkami a FooterTemplate pro spodní část ovládacího prvku.

Výhodou takového přístupu je schopnost významně ovlivnit, jak přesně chcete, aby vypadalo zobrazené uživatelské rozhraní.

```
<asp:DataList id="MujDataList" RepeatColumns="2" runat="server" >
   <ItemTemplate>
   <table cellpadding="10" style="font:10pt verdana">
     <tr>
     <td width="1" bgcolor="BD8672" />
     <td valign="top" >
     <img align="top" src='<%# DataBinder.Eval
       (Container.DataItem, ProductImageURL) %>' >
     </td>
     <td valign="top">
       <b>Jméno: </b>
       <%# DataBinder.Eval (Container.DataItem,
             ProductName) %> <br>
       <b>Popis: </b>
       <%# DataBinder.Eval (Container.DataItem,
             ProductDescription) %> <br>
       <b>Cena: </b>
       <%# DataBinder.Eval (Container.DataItem,
             ProductPrice, "{0} Kè") %> <br>
     </td>
     </tr>
   </table>
   </ItemTemlate>;
</asp:DataList>
```

Uvedený kód je relativně jednoduchý a nevyžaduje o nic více než kód připojení k datům z předchozího příkladu. Můžeme určit počet sloupců, v nichž budou data zobrazena a seznam se automaticky postará o jejich realizaci.

Validační ovládací prvky

ASP.NET validační ovládací prvky:

- Realizují komponenty, které mohou vykonat z 90, ale i více procent úloh typické kontroly hodnot na vstupech stránek.
- Provádí validaci klientským skriptem na moderních typech prohlížečů nebo zůstane u čistého HTML 3.2 a serverovou kontrolou, je-li potřeba.
- Poskytuje pružné API pro snadnou realizaci libovolné kontroly, která není poskytnuta dodávanými komponentami.

Validační ovládací prvek je visuální ASP.NET objekt, který kontroluje konkrétní podmínku platnosti jiného ovládacího prvku. Obecně se uživateli jeví jako kus textu, který se zobrazí nebo skryje v závislosti na tom, jestli validovaný prvek obsahuje chybný vstup. Může být také obrázkem nebo dokonce zcela neviditelný a přesto vykonávat užitečnou práci. Existuje pět typů validačních ovládacích prvků vykonávajících odlišné typy kontroly (vyhodnocení výrazu).

Zvláštním validačním prvkem je ValidationSummary. Stránky s velkým počtem vstupů většinou obsahují místo, kde jsou zobrazovány všechny chyby. ValidationSummary automaticky vytváří konsolidované shrnutí obsahu chyb od jednotlivých validačních ovládacích prvků na stránce.

Nakonec sám objekt stránky zveřejňuje vlastnost s názvem IsValid, kterou lze testovat v serverovém kódu, aby jste zjistili platnost uživatelského vstupu. Vždy by jste měli platnost vstupních dat kontrolovat i na serveru, neboť by mohlo být celkem snadné zlomyslně modifikovat HTML a obejít klientskou validaci.

webové XML služby

Infrastruktura ASP.NET web služeb poskytuje mnoho výhod, které zjednodušují vývoj a používá programový model dobře známý vývojářům pracujícím s ASP nebo Visual Basic. Nemusíte porozumět všem detailům HTTP, SOAP, WSDL nebo jiným specifikací webových služeb, aby jste model mohli používat.

Klienti mohou zasílat dotazy přes SOAP, HTTP GET a HTTP POST. Jsou definovány standardy pro kódování metod a parametrů jako dotazovacích řetězců pro HTTP GET a data formulářů pro HTTP POST. HTTP GET a HTTP POST mechanismy nejsou tak výkonné jako SOAP, jelikož neumí prezentovat složité datové typy, ale zpřístupňují web služby i klientům, kteří nepodporují SOAP.

Model ASP.NET web služeb předpokládá nestavovou architekturu. Nestavová architektura je obecně lépe škálovatelná ve srovnání s plně stavovou. Web služby mohou používat služby managementu stavu ASP.NET v případě, že potřebují stav uchovat mezi jednotlivými dotazy. Web služby založené na platformě ASP.NET jsou řízené aplikace běžící v CLR a mohou vyžívat všech dříve diskutovaných výhod a vlastností CLR a .NET Framework.

ASP.NET také dodává SDK nástroj (wsdl.exe) sloužící ke generování silně typového řízeného prostředníka (proxy) webové služby popsaného pomocí Web Service Description Language (WSDL) souboru. Proxy generátor mapuje zprávy popsané ve WSDL souboru na metody vytvořené třídy. Proxy zakrývá ohromnou část síťové infrastruktury a způsobu předávání dat. Použití webové služby proto vypadá podobně jako použití jakékoli jiné řízené třídy. Implementace proxy preferuje SOAP při komunikaci s Web službou, ale podporuje i mechanismy HTTP GET a HTTP POST. Při vývoji webové služby píšete místo souboru .aspx soubor s příponou .asmx, který obsahuje třídu s libovolnými metodami, jenž chcete zveřejnit. Když se prohlížečem odkážete na .asmx soubor, dojde k jeho překladu, je generován WSDL kontrakt, včetně implicitní stránky, která službu dokumentuje a poskytuje jednoduchý způsob, jak službu testovat.

Web služby umožňují zveřejnit zákaznickou obchodní funkcionalitu (jako např. informace o dopravě zásilky) prostřednictvím Webu. Pro tento účel napíšeme třídu, která zpřístupňuje své metody jako URI, který vrací data v XML formátu. Kupříkladu by mohla jedna společnost volat dodávkovou službu jiné společnosti a zahrnout její výsledky do své vlastní aplikace pro sledování pohybu dodávek. Zde je ukázka, jak by mohla dodávková společnost realizovat Webovou službu v jazyce C#:

```
<%@ WebService language="c#" Class="Dodavky" %>
using System.Web.Services;
public class Dodavky
{
   [WebMethod]
   public string StavObjednavky(string CisloObjednavky) {
     // kód pro naètení podrobností z datového skladu
     return Stav;
   }
}
```

Uvedený kód by mohl být uložen do souboru pod názvem Sledovani.asmx do aplikačního adresáře na webovém portálu dodávkové společnosti. Společnost klienta by pak mohla tuto webovou služby vyvolat několika různými způsoby. Mohli by použít HTTP-GET, kde by byly parametry metody předány jako součást řetězce dotazu. Například:
http://objednavky.ups.com/objednavky/Sledovani.asmx/StavObjednavky?CisloObjednavky=BRU123

Společnost klienta může použít HTTP-POST, kde jsou parametry metody předány uvnitř těla odeslaného dotazu jako hodnoty formuláře. Nebo může použít HTTP-SOAP, kde parametry metod jsou zabaleny do průmyslového standardu XML.

Nyní se může výsledný zákazník dotázat u společnosti klienta (jeho poskytovatele) na podrobnosti dodávky a webové aplikace klientské společnosti shromáždí informace o pozici zboží od dodávkové společnosti a vrátí je zpět společně se stavem produktu. Společnost klienta může také zveřejnit své vlastní podrobnosti o stavu objednávky jako webovou službu a umožnit její konzumaci jiným zákazníkům.

Báječnou vlastností webových služeb je, že umožňují zveřejnit službu bez nutnosti zveřejnění dat nebo interních obchodních pravidel. Zatímco automaticky poskytujete obchodní služby, vaše data a kód jsou bezpečně ukryty.

Vylepšená technologie Cache
Pro vybudování vysoce výkonného a škálovatelného portálu je bezpodmínečně nutný nějaký způsob cachování. Jisté operace při vytváření stránky jsou velice náročné a jejich umístění do cache ve chvíli, kdy jsou vytvořeny, ušetří systémové zdroje a umožní výrazně rychlejší poskytování. ASP.NET nabízí následující:

- Caching na úrovni stránek, umožňující umístění celé stránky do cache
- Fragmentový caching, který umožňuje uložení částí stránek do cache
- Cache API, které programátorům zpřístupňuje jádro systému cache

Caching celých stránek je nejjednodušším typem a umožňuje umístění dynamických stránek do výstupní cache. Tím dosáhneme přímé obsluhy z cache místo opakovaného spouštění při každém individuálním dotazu. Lze určit jednak absolutní čas (například půlnoc), jednak relativní čas (například 20 minut poté, co se na stránku naposledy přistoupilo) a jemně řídit, jak dlouho stránky v cache setrvají. Caching celých stránek je citlivý na URL požadavek a řetězec dotazu. Když se ke stránce přistoupí s jistými parametry, dojde k navrácení cache verze pouze tehdy, když jsou parametry shodné. Fragmentový caching nabízí stejnou úroveň kontroly nad jednotlivými částmi ASP.NET stránek a umožňuje uložení určitých částí stránek do cache, zatímco ostatní části se nadále dynamicky generují.

Obě uvedené techniky používají Cache API, které je dostupné i programátorům a umožňuje jim ukládat do cache své vlastní objekty a přistupovat k nim místo opakovaného zakládání pokaždé, když jsou třeba. Cache API obsahuje expirační pravidla, stejně jako upozornění na změny v souborech, které vám umožní inteligentní využití cache pro významné navýšení výkonnosti.

Cache je speciálně výhodná u portálů zpřístupňující katalogy produktů, kde obsah musí být dynamický, ale nemění se příliš často. Pokud potřebujete dynamický katalog, v ASP jej musíte buď generovat z databáze při každém dotazu na stránku, nebo jej převést na HTML. Díky stránkové cache v ASP.NET můžete specifikovat, že konkrétní dynamická stránka má být v cache celý den, a tak každý den, při prvním přístupu, je její obsah do cache uložen.

Ladění

ASP.NET je založeno na CLR a jako důsledek můžete bezproblémově ladit kód na ASP.NET stránkách, přecházet do metod objektů a vracet se zpět. Díky způsobu, jakým jsou ASP.NET stránky překládány, se do nich může.NET Framework za běhu napíchnout. Realizuje je se tak velmi bohaté jazykově nezávislé ladící prostředí. Jednoduše lze krokovat z vaší webové stránky napsané v jazyce Visual Basic.NET do ovládacího prvku naprogramovaného v jazyce C#.

Trasování

ASP.NET podporuje trasování jak na úrovni stránek, tak na úrovni aplikace. S podporou ASP.NET pro trasování stránka sbírá všechny podrobnosti o dotazu na stránku, včetně stromu serverových ovládacích prvků, serverových proměnných, hlaviček, cookies a parametrů z řetězce dotazu/formuláře a umísťuje je do formátované tabulky do dolní části stránky. Obsahuje také uložené informace ze všech událostí vyvolaných na stránce, včetně jejich přesného času.

Trasování je ASP.NET realizováno formou metody Trace objektu stránky. Například, pro trasování vstupu a výstupu z procedury události můžete použít následující kód:

```
Sub Page_Load (Source As Object, E As EventArgs)
   Trace.Write   Page_Load, Start
   ' Zbytek kódu bude zde
   Trace.Write   Page_Load, End
End Sub
```

Pro povolení výstupu této trasovací informace použijete direktivu Page:

```
<%@ Page Trace="True" %>
```

Vyžadujete-li trasovací informace, ale nepřejete si, aby je uživatelé viděli, můžete vždy nastavit aplikaci tak, aby se trasovací informace posílaly na jiné místo, odkud mohou být prozkoumány později.

Nasazení
ASP.NET používá Microsoft .NET Framework model nasazení, založený na principu assembly. Ve výsledku lze těžit z vlastností jako je xcopy nasazení, souběžné nasazení assembly a konfiguraci formou XML.

ASP.NET také podporuje nasazení plně přeložených aplikací. Výhodou je skutečnost, že žádný ze zdrojových souborů není přístupný administrátorovi webového serveru. Důležitá vlastnost, pokud jiná společnost provozuje vaši aplikaci.

ASP.NET obsahuje extrémně jednoduchý model nasazení. Pro nasazení aplikace jednoduše překopírujete všechny soubory, z nichž se sestává, do odpovídajícího adresáře. Není nutná žádná registrace objektů nebo restartování aplikace. Všechny části ASP.NET aplikace mohou být nasazeny tímto způsobem, včetně .aspx stránek, webových služeb, přeložených komponent (uložených v DLL souborech) a dokonce konfiguračních dat.

Pro přeložené komponenty nyní existuje speciální adresář „bin“, umístěný přímo v aplikačním kořeni. Jakákoliv komponenta umístěná do tohoto adresáře je automaticky dostupná pro přímé použití v aplikaci. Jelikož jsou dostupné pouze této konkrétní aplikaci, dochází opravdu ke skutečné aplikační izolaci. Každá aplikace na serveru může běžet s mírně odlišnou verzí jisté komponenty bez jakéhokoliv vlivu na další aplikace. Přináší to ohromnou výhodu vývojářům, kteří provozují na jednom serveru více instancí jednoho portálu, kam patří vývojové prostředí, testovací prostředí a produkční prostředí, každé s odlišnou verzí komponent.

ASP.NET již nezamyká DLL knihovny v paměti a nebrání vám před jejich výměnou. Nyní jednoduše zkopírujete novou DLL přes existující. ASP.NET automaticky detekuje změny a přejde na novou verzi. Pro rozpracované dotazy nadále používá starou verzi, zatímco nové dotazy jsou přesměrovány na verzi novou. Již nikdy nemusíte znovu zavádět nebo restartovat aplikaci. Pro deinstalaci celé aplikace nebo části aplikace jednoduše smažete odpovídající soubory. Již nemusíte deregistrovat komponenty a odstraňovat položky v Registry.

Škálovatelnost a dostupnost
ASP.NET automaticky zpracovává jak chyby uživatelského, tak systémového kódu. Automaticky se také zotaví z porušení ochrany přístupu, úniků paměti a deadlocků.

ASP.NET vám umožňuje konfiguraci maximálního počtu dotazů, které mohou být umístěny do fronty zpracovávané aplikací před tím, než je nastartován nový proces. V tomto případě jsou události automaticky přiřazeny ke zpracování novému procesu. Obdobně můžete konfigurovat maximální hodnotu paměti zabírané aplikací. Jestliže dojde k překročení tohoto maxima, je nastartován nový proces a požadavky jsou opět k němu přesměrovány. V obou případech je starý proces ukončen, jakmile dokončil zpracování všech čekajících dotazů.

ASP.NET navíc podporuje koncepci preventivní recyklace aplikací. Eliminuje se tím potřeba restartovat aplikaci jednou za týden, abychom zajistili, že naše aplikace skutečně běží. ASP.NET může restartovat ASP.NET proces na základě počtu dotazů obsloužených během doby běhu aplikace.

Aplikační model Windows Forms

Vývojáři, vytvářející klientské aplikace pro Windows, mohou použít aplikační model Windows Forms a získat výhodu veškerého pokročilého uživatelského rozhraní poskytovaného Windows, včetně existujících COM ovládacích prvků a nových vlastností ve Windows 2000, jako jsou průhledná, vrstvená nebo plovoucí okna. Lze si vybrat mezi tradičním Windows nebo web vzhledem. Programový model Windows Forms a podporu návrhu shledáte jako vývojáři velice intuitivní, díky své podobnosti s existujícími prostředky pro tvorbu formulářových aplikací.

Windows Forms jsou součástí .NET platformy a používají mnoho nových technologií, včetně společného aplikačního frameworku, řízeného exekučního prostředí, integrované bezpečnosti a principů objektově orientovaného návrhu.

Windows Forms vám navíc umožňují konzumovat webové služby a vytvářet bohaté, databázově orientované aplikace s datovým modelem ADO.NET. Díky novému sdílenému vývojovému prostředí ve Visual Studiu.NET můžete vytvářet Windows Forms aplikace pomocí všech jazyků, které podporují .NET platformu.

Visuální dědičnost

Visuální dědičnost je jednou z klíčových vlastností dostupných ve Windows Forms, která zvyšuje produktivitu vývojářů a umožňuje opakované požití kódu. Organizace může například definovat standard základního formuláře, který bude obsahovat položky jako jsou firemní logo a třeba společnou nástrojovou lištu. Tento formulář může být vývojáři použit formou dědičnosti a dále rozšířen podle požadavků konkrétní aplikace, při zachování společného uživatelského rozhraní napříč organizací. Tvůrce základního formuláře může určit, které jeho elementy mohou být modifikovány, rozšířeny a které musí zůstat beze změn a zajistit, že formulář je korektně používán. Tato funkcionalita je velice dobře známá Visual FoxPro vývojářům.

Přesný návrh formuláře

Vývojáři mají k dispozici nevídanou úroveň řízení a produktivity při návrhu vzhledu Windows Forms aplikací. Vlastnosti jako In-Place Menu Editor, ukotvení ovládacích prvků, uchycení prvků a mnoho nových ovládacích prvků nabízí vývojářům vyšší úroveň výkonnosti a přesnosti při vývoji bohatého uživatelského prostředí Windows aplikací.

Pomocí In-Place Menu Editoru mohou vývojáři rychle a jednoduše přidávat menu na formuláře, modifikovat je a ověřovat jejich vzhled, bez nutnosti spouštět aplikaci. Ovládací prvky na formuláři jsou mnohem efektnější díky možnostem ukotvení, které automaticky přizpůsobují jejich velikosti rozměru formuláře. Díky technice uchycení mohou být prvky automaticky přichyceny k libovolné straně formuláře a poskytnout velkou pružnost jeho návrhu.

Také existující COM komponenty lze používat a spouštět na formulářích a zachovat tak investice do existujících technologií.

Nové ovládací prvky (včetně Link Label, Tray Icon a Print Preview) nabízejí vývojářům další běžnou funkcionalitu. Link Label realizuje odkaz ve stylu HTML na zadaný URL. Text zobrazený tímto ovládacím prvkem bude podtržen, kurzor myši se změní na tvar ruky, jakmile je myš nad něj přesunuta, a po kliknutí dojte k vyvolání události. Tray Icon umožňuje vývojářům vytvářet aplikace, které běží ve Windows Tray, obdobně jako Microsoft SQL Server Service Manager. Windows Forms nabízí celou tiskovou infrastrukturu, která zjednodušuje realizaci tisku, včetně okna náhledu před tiskem ve formě ovládacího prvku Print Preview.

Vývojáři mohou vytvářet s Windows Forms aplikace, které oslovují mnohem širší skupinu uživatelů. Ovládací prvky Windows Forms implementují programové rozhraní Microsoft Active Accessibility, usnadňující vytváření aplikací, které podporují pomůcky jako jsou čtečky obrazovek.

Nižší TCO (Total Cost of Ownership)
Windows Forms nabízejí mnohem více než ohromný způsob, jak vytvářet plnohodnotné Windows aplikace. Můžete také těžit ze schopností jednoduchého nasazení a integrovaného aplikačního bezpečnostního modelu. Windows Forms využívají výhody spojené s řízením verzí a schopnosti nasazení na Microsoft .NET platformě pro snížení nákladů spojených s nasazením, včetně vyšší aplikační robustnosti v dlouhodobém horizontu. Dochází tak k významnému snížení nákladů na údržbu (TCO) aplikací napsaných pomocí Windows Forms.

Při realizaci Windows Forms aplikací nemáme potřebu nasazovat aplikace na osobní počítače koncových uživatelů. Uživatel může aplikace jednoduše spustit jednoduchým zapsáním URL v prohlížeči. Aplikace bude stažena na klientský počítač, spuštěna v bezpečném exekučním prostředí a automaticky se odstraní po svém ukončení.

Organizace, které chtějí fyzicky nasadit aplikace na koncová pracoviště, nemusí provádět instalační proces náročný na systémové zdroje. Uživatelé nebo administrátoři jednoduše překopírují aplikace na odpovídající počítač.

Visual Studio.NET

Visual Studio.NET obsahuje významná vylepšení v uživatelském rozhraní: ukotvitelné a zajíždějící nástrojové lišty. Tyto lišty zajíždějí na stranách obrazovky do formy záložek do chvíle, kdy je potřebujete použít a poskytují vám mnohem více místa na pracovní ploše.

Dalším vylepšením je společné Integrované Vývojové Prostředí (Integrated Development Environment - IDE) pro všechny jazyky. Mezi nejvíce zajímavé vlastnosti patří:

- Schopnost minimalizovat a plně zobrazovat procedury za účelem skrytí kódu, který nepotřebujete dočasně vidět v době, kdy pracujete na jeho jiné části.
- Máte-li otevřeno několik projektů, které mohou být napsány v různých jazycích, IDE automaticky přepíná prostředí, podle aktuálně zvoleného.
- Seznam úkolů zobrazuje všechny chyby překladu společně s uživatelsky definovaným poznámkami. Dvojklikem na položku seznamu „TODO:“ jste automaticky přeneseni na místo v kódu.
- Automatické odsazování je rozšířeno o automatické zarovnávání řídících struktur.

Další výhodou společného prostředí je sjednocená množina vizuálních návrhářů, včetně návrhářů HTML, XML, dat a kódu pro serverovou stranu.

Visual Studio.NET obsahuje jeden unifikovaný projektový model. Tento nový model vám umožní integraci komponent z VB.NET, C# i Managed Extensions for C++ do jednoho projektu, na kterém pracuje celý váš tým. Ve výsledném efektu je projekt schopen zahrnout komponenty vytvořené v jazyce, který je pro ně nejvhodnější.

Dynamický help poskytuje další hodnotné rozšíření. IDE dobře ví, kde se právě pohybujete a jakou úlohu právě řešíte. Dynamický help vám pomůže získat nové znalosti a dokončit výrazy obdobným způsobem, jakým vám poskytuje IntelliSense informace o metodách a vlastnostech. Systém helpu vám poskytne nápovědu nejen ve stylu slovníku (jako vlastnost IntelliSense), ale skýtá také vysokoúrovňové programátorské rady na téma, jak nástroj správně používat.

Jazyky

Všechny .NET jazyky využívají .NET Common Laguage Runtime (CLR) a Base Class Library (BCL) implementací společné množiny minimálních vlastností definovaných v Common Language Specification (CLS). Každý jazyk také podporuje Common Type System (CTS), umožňující jejich vzájemnou kompatibilitu. Zatímco některé jazyky nativně podporují CTS, jiné, jako Managed C++, realizují podporu pomocí rozšíření základního jazyka.

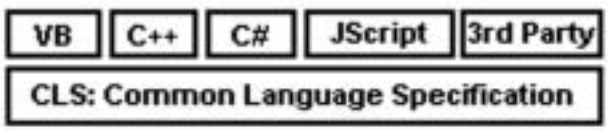
Jedním z principů návrhu jazyka Visual Basic vedoucím ke zjednodušení programování je například ukrytí všech operací s pamětí před vývojářem. Jde o naprosto opačný přístup než u jazyka C++. Zatímco nové schopnosti Garbage Collection (GC) poskytované .NET Framework jsou přirozeně podporovány ve Visual Basicu, C++ vyžaduje modifikace a rozšíření, která jsou dostupná v Managed C++.

Výběr jazyka

Uvnitř CLR všechny jazyky sdílí rozsáhlou množinu zdrojů, včetně:

- Objektově orientovaného programového modelu (dědičnost, polymorfismus, zpracování výjimek a garbage collection)
- Bezpečnostního modelu
- Typového systému
- Base Class Library (BCL)
- Vývojových, ladících a profilovacích nástrojů
- Řízení běhu aplikace a kódu
- Překladače MSIL-nativní kód a optimalizérů

Common Language Specification (CLS) je společnou podmnožinou jazyků se Společným Typovým Systémem (Common Type System - CTS), zanesenou jako sada pravidel. Jazyky, které vyhovují těmto pravidlům mohou navzájem plně spolupracovat. Můžete například založit základní třídu v jazyce C# a odvodit z ní novou třídu v jazyce Visual Basic.NET.



Obrázek 19 CLS a jazyky

Kód pro platformu .NET můžete psát v jazyce Microsoft Visual C++. C++ kód však nemůže být přímo řízen mechanismy CLR, protože nevyhovuje specifikaci CLS. Proto Microsoft přidal do C++ množinu řízených rozšíření a kód, který s nimi pracuje, již CLR vyhovuje.

Vysoká úroveň spolupráce jazyků ze základu ovlivňuje způsob, jakým si vybíráte programovací jazyky a implementujete své návrhy. Výběr jazyka se stává více osobní

záležitostí a závisí na syntaxi, která vám nejvíce vyhovuje. CLR také může posílit širší přijetí méně rozšířených jazyků, protože budou mít stejné možnosti jako jazyky tvořící hlavní proud.

Rozdíly v oblasti výkonnosti a možnostech jsou mezi jazyky Visual Basic.NET a C# znatelně menší než tomu bylo u předchozích verzí Visual Basic a Visual C++. Například:

- Architektury obou jazyků VB.NET a C# jsou od počátku postavena nad .NET Framework
- VB.NET a C# mají velice podobné možnosti ve smyslu programovacího výkonu. Rozdíly jsou v funkcionalitě založené na potřebách případných Visual Basic a C# vývojářů, kdy například C# podporuje vložení neřízeného C++ kódu.
- VB.NET a C# jsou však významně odlišné pokud jde o zkušenosti uživatelů. Např., citlivost na výšku znaků je základní charakteristikou C++ a promítá se i do C#. Tento jazykový aspekt však má za následek ohromný rozdíl ve zkušenostech Visual Basic programátora, který se snaží naučit C# nebo C++.

Visual Basic.NET

Visual Basic.NET přináší nemalý počet významných vylepšení oproti předchozím verzím Visual Basicu, zejména díky základu v typovém systému .NET. Hlavní vylepšení se vztahují k podpoře objektově orientovaných principů jako polymorfismus, dědičnost a přetížení operátorů. Ostatní vycházejí z jeho společného základu poskytovaného CLR, stejně jako u ostatních jazyků. Můžete volně předávat datové typy mezi komponentami vyvinutými v jiných programovacích jazycích a dědit ze základních tříd taktéž definovaných v jiných jazycích. IntelliSense funguje na libovolných komponentách, nezávisle na vývojovém jazyce. Visual Basic.NET podporuje přístup k datům pomocí ADO.NET, programový model Windows Forms, multithreading a nové vylepšené průvodce a návrháře.

Stávající vývojáři v jazyce Visual Basic budou mít prospěch z nových jazykových konstrukcí a vlastností, které jim pomohou při tvorbě mnohem výkonnějších a robustních aplikací. Nemusíte však přímo přepisovat všechen existující kód, protože kód z Visual Basic 6 a kód z Visual Basic.NET poběží současně bez problémů. Pokud se rozhodnete pro náhradu, služby COM Interoperability vám významně pomohou překonat přechodnou fázi tím, že umožní snadné využití existujících komponent. Visual Basic.NET také obsahuje Upgrade Wizard, který vás krok po kroku provede upgrade procesem a zajistí nezbytné kontroly změn jazykové syntaxe v době, kdy je nahráván Visual Basic 6 projekt. Více informací o převodu aplikací z Visual Basic 6 najdete na adrese ***http://msdn.microsoft.com/library/techart/vb6tovbdotnet.htm.***

Základní nové vlastnosti uvedené ve Visual Basic.NET jsou:

- Dědičnost. Třídy mohou být rozšířeny (neboli odvozeny) z existujících tříd, díky implementační dědičnosti. Visual Basic.NET (společně se všemi ostatními .NET jazyky) podporuje pouze jednoduchou dědičnost, což znamená, že můžeme dědit pouze z jedné základní třídy. Nová třída může získat funkcionalitu a chování třídy základní a vybírá si, kde chování modifikuje. Odvozená třída může také přidat nové chování, například implementací nové metody. Abychom mohli použít implementační dědičnost, musíme novou třídu deklarovat pomocí klíčového slova Inherits a vyjádřit, ze které základní třídy bude nová odvozena. Dále můžete použít klíčové slovo Overrides pro modifikaci funkcionality základní třídy, kdekoli je to potřeba. Všimněte si, že metoda, která má mít schopnost předefinování v odvozené třídě (s klíčovým slovem Overrides), musí být v základní třídě označena slovem Overridable. Ukázka je uvedena v následujícím příkladu:

```
Public Class class1
  Overridable Function Jmeno () As String
  ' sem pøijde implementace
  End Function
End Class

Public Class class2
  Inherits class1
  Overrides Function Jmeno () As String
  ' implementace pøedefinované funkce
  End Function
End Class
```

- Polymorfismus. Jde o schopnost chovat se k objektům různých typů tříd stejným způsobem. Tradičně jsme toho mohli dosáhnout ve Visual Basicu pomocí rozhraní. Visual Basic.NET stále podporuje polymorfismus formou rozhraní, ale můžeme jej dosáhnout i pomocí dědičnosti.
- Konstruktory. Konstruktory umožňují, aby byly objekty zakládány a inicializovány na počáteční stav v jedné operaci. Tím odstraníme potřebu založit objekt a následovně volat speciální metodu pro jeho inicializaci. Konstruktory objektů ve Visual Basic.NET umožňují předávat objektům hodnoty, ve chvíli, kdy jsou zakládány. Dochází ke zjednodušení kódu a eliminaci chyb při práci s objekty.

```
' založení a inicializace nového objektu Clovek
objClovek = New Clovek ( Adam, Novak, 1996)
```

- Inicializátory. Inicializátory umožňují na jednom řádku kódu deklarovat proměnnou a naplnit ji počáteční hodnotou. Vede to na menší, jednodušší a lépe udržitelný kód:

```
Private mProp As Int32 = 72
Dim sNazev As String = Marketing
Dim dblPomery () As Double = {5.3, 55.8, 82.2}
```

- Strukturované zpracování výjimek. .NET CLR používá výjimky pro jednotné zpracování chybových stavů, nezávisle na programovacím jazyce, který používáte. Výjimky jsou odpovědí na chybný předpoklad, který nastane za běhu aplikace. Mohou být jednak generovány uvnitř .NET runtime (systémové výjimky), jednak být vytvořeny programově (aplikační výjimky). Výjimky jsou propagovány přes jazykové hranice. Například komponenta napsaná v jazyce C# může generovat výjimku, která může být následovně zachycena a zpracována klientským programem vyvinutým v jazyce Visual Basic.NET. Následující kód z webového portálu Microsoft ukazuje syntaxi zpracování výjimek v jazyce Visual Basic.NET.

```
Sub SEH()
  Try
        Open TESTFILE For Output As #1
        Write #1, ZakaznickeInformace
  Catch
        Kill TESTFILE
  Finally
        Close #1
  End Try
End Sub
```

- Konstrukce Try...Catch...Finally výrazně zjednodušují umístění zpracování chyby do blízkosti kódu, který by mohl způsobit chybu.

C# (C Sharp)

Během minulých dvou desetiletí byly jazyky C a C++ nejčastěji používanými nástroji pro vývoj komerčního a obchodního software. Zatímco oba jazyky poskytují vývojářům obrovské množství jemně členěného řízení, tato pružnost přichází na úkor produktivity. Ve srovnání s jazykem jako je Visual Basic, trvá vývoj ekvivalentních aplikací v C a C++ mnohem déle. Kvůli složitosti a dlouhému vývojovému cyklu spojenému s těmito jazyky, hledalo mnoho programátorů jazyk, který by nabídnul lepší poměr mezi výkonem a produktivitou.

V současnosti existují jazyky, které zvyšují produktivitu, ale obětují pružnost, kterou programátoři v C a C++ většinou vyžadují. Taková řešení vývojáře příliš omezují, například vynecháním mechanismu na nízkoúrovňovém řízení kódu, a poskytují pouze vlastnosti nacházející se ve společném průsečíku jazyků. Velice špatně spolupracují se staršími systémy a často je lze velmi obtížně kombinovat se současnými programovými praktikami na Webu.

Ideálním řešením pro C a C++ programátory je rychlý vývoj kombinovaný s výkonným přístupem k veškeré funkcionalitě základní platformy. Chtějí prostředí zcela synchronizované s objevujícími se webovými standardy a dále se systémy, které realizují integraci s existujícími aplikacemi. C a C++ programátoři by navíc přivítali schopnost kódu pracovat na velmi nízké úrovni, pakliže je potřeba.

C# je odpovědí firmy Microsoft na popsaný problém. C# je moderním, objektově orientovaným jazykem, který umožňuje programátorům rychlé vytváření širokého rozsahu aplikací pro Microsoft .NET platformu. Díky svému elegantnímu objektově orientovanému návrhu je C# dobrou volbou pro vytváření velkého množství různých komponent, počínaje vysokoúrovňovými obchodními objekty až po aplikace na systémové úrovni. Díky podpoře na .NET platformě, mohou být komponenty převedeny na webové služby, které je možné vyvolávat přes Internet z různých

jazyků, běžících na libovolných operačních systémech. C# je navržen tak, aby poskytl C++ programátorům zrychlený vývoj bez obětování výkonnosti a řízení, které byly charakteristické pro jazyky C a C++. C# je do vysoké míry podobný jazykům C a C++. Vývojáři znalí těchto jazyků mohou začít v C# velice rychle programovat.

- Produktivita a bezpečnost. Nový typ ekonomiky tlačí na průmysl, aby reagoval na konkurenční hrozby mnohem rychleji než kdykoli předtím a vývojáři jsou nuceni, aby zkrátili doby vývojových cyklů a vytvářeli více inkrementálních oprav programů, raději než samostatné velké verze. C# byl navržen s těmito vizemi. Jazyk je vytvořen tak, aby pomohl vývojářům dosáhnout lepšího výsledku s nižším počtem řádků kódu a menší příležitostí vytvořit chybu. Dokonce experti v C++ mohou vytvořit ty nejprimitivnější chyby. Například zapomenou inicializovat proměnnou, nebo neuvolní systémový zdroj. Tyto chyby velmi často vyústí v neočekávané problémy, které zůstávají skryté po dlouho dobu. Jakmile je aplikace uvedena do produkce, může být velice nákladné opravit sebemenší programovou chybu. Moderní návrh jazyka C# odstraňuje většinu běžných programových chyb v C++. Například:
  - Garbage collection oprošťuje programátory od údělu manuálního řízení paměti.
  - Členské proměnné v C# jsou automaticky inicializovány prostředím.
  - Jsou zakázána nebezpečná přetypování.
- Konečným výsledkem je jazyk, který vývojářům významně ulehčí úkol tvorby a údržby programů, řešících složité obchodní problémy. Byl také navržen s cílem snížit průběžné náklady na vývoj a dále s integrovanou podporou verzování. Proces náhrady softwarových komponent je vždy náchylný k chybám. Změny provedené v kódu mohou nechtěně změnit sémantiku exitujících programů. C# obsahuje přímo v jazyce podporu řízení verzí a snaží se vývojářům pomoci s řešením tohoto problému. Odpovídající vlastností je podpora rozhraní a dědičnosti rozhraní. Tyto vlastnosti umožňují vývoj a rozvoj složitých systémů v čase. Proces vývoje pozdějších verzí projektu je díky uvedeným vlastnostem mnohem stabilnější a snižuje celkové vývojové náklady následujících verzí.
- Výkon, výrazovost a pružnost. C# nabízí lepší propojení mezi obchodním procesem a implementací. Vzhledem k vysoké úrovni vynaloženého úsilí, které společnosti vkládají do obchodního plánování, je naprosto nezbytné, aby existovalo úzké spojení mezi abstraktním obchodním procesem a vlastní softwarovou implementací. Většina jazyků například neobsahuje možnost jednoduchého spojení informací o návrhu s vlastním kódem. Vývojáři často v současnosti používají komentáře kódu k určení, které třídy tvoří jistou část abstraktního obchodního objektu.
  .NET umožňuje architektům projektu definovat doménově specifické atributy a aplikovat je na libovolný element jazyka, například třídy, rozhraní a podobně. Vývojář může následovně programově analyzovat atributy jednotlivých elementů. Významně se tak například ulehčí vytvoření automatického nástroje, který zajistí, že každá třída nebo rozhraní je odpovídajícím způsobem označeno jako část konkrétního abstraktního obchodního objektu, nebo jednoduše vytvoří zprávu založenou na doménově specifických atributech objektů. Úzká vazba mezi zákaznickými metadaty a kódem programu napomáhá posílit spojení mezi zamýšlením chováním programu a jeho vlastní implementací.
  C# také nabízí velmi široké možnosti spolupráce. Řízené a typově bezpečné prostředí je vhodné pro většinu podnikových aplikací, ale každodenní zkušenosti ukazují, že některé aplikace stále vyžadují „nativní“ kód buď z výkonnostních důvodů, nebo z důvodů spolupráce s programovými rozhraními existujících aplikací (API). Takové scénáře mohou přinutit vývojáře

používat C++, i když by upřednostnili mnohem produktivnější vývojové prostředí.

- Interoperabilita. Jelikož je C# postaven na .NET platformě, každý C# objekt může být použit jako by byl COM objektem. Vývojáři již nemusí nadále explicitně implementovat IUnknown a další COM rozhraní, jak tomu bylo při realizaci COM komponent v jiných jazycích. Tyto vlastnosti jsou nyní interní součástí. Programy napsané v C# mohou obdobně využívat existující COM objekty, nezávisle na tom, kdo je vytvořil.
- Přístup k platformě. C# obsahuje speciální vlastnosti pro ty, kteří mají zájem, aby jejich programy mohly volat libovolné funkce nativního API v jazyce C. Vývojářům je povoleno uvnitř speciálně označených bloků kódu používat ukazatele a tradiční vlastnosti C/C++ jako jsou manuální řízení paměti a ukazatelová aritmetika. Jde o ohromnou výhodu, kterou má jazyk C# před ostatními prostředími. Znamená to, že C# programátoři mohou stavět na existujících C a C++ základech formou Managed Extensions to C++, místo nechtěného vyhození existujícího kódu.
- Standardizace C#. Ve chvíli, kdy Microsoft dokončil referenční popis jazyka C#, zveřejnil současně snahu o jeho standardizaci. C# byl formálně předán jako návrh k organizaci ECMA Technical Committee (TC) 39. Jde o stejný výbor, který standardizuje ECMAScript (JScript).
  V případě, že členové výboru schválí návrh a budou souhlasit se standardizací C#, další výrobci mimo Microsoft budou smět volně jazyk implementovat. Samozřejmě můžete najít výrobce implementující jak „standardní C#“, tak jejich vlastní C# rozšíření, obdobně jako dnes implementují standard jazyků C a C++ společně s jejich vlastními extenzemi. V tomto smyslu se očekává, že případný C# standard bude napodobovat rovnováhu mezi přenositelným chováním a specifickými rozšířeními, jak je tomu u současných C a C++ standardů.
- Skutečný komponentově orientovaný vývojový jazyk. C# zjednodušuje vývoj komponent. Komponentové koncepce jako vlastnosti, události, řízení verzí a metadata jsou prvořadými vlastnostmi zabudovanými do jazyka. Ve výsledném efektu došlo k eliminaci externích hlavičkových a IDL souborů, společně s přiřazením GUID a definic rozhraní. Nepotřebujeme již ani speciálně pojmenované metody používané k vytvoření iluze přítomnosti vlastností. Komponenty by také měly být samopopisné. Primárně k tomu dochází pomocí meta-dat, která nemohou být oddělena od komponent (jako součást komponentní assembly). Programátor si může také vložit do zdrojového kódu komentáře ve formátu XML tagů. Z tagů ve zdrojovém kódu lze později překladačem generovat dokumentaci s XML syntaxí.

Řízený C++.NET

Managed Extensions for C++ jsou množinou jazykových rozšíření, které pomáhají vývojářům v Microsoft Visual C++® psát aplikace pro platformu Microsoft .NET.

Řízená rozšíření jsou užitečná pokud:

- Chcete po fázích migrovat velkou část kódu z neřízeného C++ na .NET platformu
- Máte neřízené C++ komponenty, které chcete používat z .NET Framework aplikací
- Máte .NET Framework komponenty, které chcete používat z neřízeného C++
- Chcete míchat neřízený C++ kód a .NET kód ve stejné aplikaci

Řízená rozšíření pro C++ poskytují vývojářům na .NET platformě nepřekonatelnou pružnost. Ve jedné aplikace lze neomezeně míchat tradiční neřízený C++ a řízený C++ kód. Nové aplikace napsané s řízenými rozšířeními mohou využít výhod z obou těchto světů. Existující komponenty lze jednoduše zaobalit pomocí řízených rozšíření do .NET komponent a při integraci s .NET zachovat investice do existujícího kódu.

Co to jsou řízená rozšíření?
Rozšíření vám umožní vytvářet řízené třídy v C++, které běží pod dohledem .NET Framework. (Neřízené C++ třídy běží v tradičním prostředí Microsoft Windows®.) Řízená třída je nativní .NET třída a může plně využívat .NET Framework.

Řízená rozšíření jsou nová klíčová slova a atributy ve vývojovém systému Visual C++. Umožní vám rozhodnout, které třídy a funkce budou přeloženy jako řízený a které jako neřízený kód. Tyto části spolu navzájem bezproblémově spolupracují, stejně jako s externími knihovnami.

Řízená rozšíření jsou také používána pro vyjádření .NET typů a koncepcí přímo ve zdrojovém C++ kódu. Umožní se tak vývojářům bezproblémová tvorba .NET aplikací, aniž by museli psát zvláštní kód.

Klíčové scénáře

- Hladká integrace existujícího kódu s .NET
  Pokud jste vložili velké investice do C++ kódu, řízená rozšíření vám pomohou provést hladký přechod na .NET platformu. Protože můžete míchat neřízený a řízený kód ve stejné aplikaci (dokonce ve stejném souboru), můžete kód postupně převést, komponentu po komponentě, na .NET. Můžete také pokračovat v psaní komponent v neřízeném C++, využívat výhody plného výkonu a pružnosti jazyka a řízená rozšíření nasadit pouze pro vytvoření tenkého, vysoce výkonného obalu, který umožní vyvolání vašeho C++ kódu z .NET komponent.
- Přístup k C++ komponentám z .NET jazyků
  Řízená rozšíření vám umožní volat C++ třídy z libovolného .NET jazyka. Musíte napsat jednoduchou třídu obalu pomocí rozšíření, která zveřejní vaše C++ třídy a metody jako řízené třídy. Obal je řízenou třídou a může být volán z libovolného .NET jazyka. Třída obalu vystupuje jako mapovací vrstva mezi řízenou třídou a neřízenou C++ třídou. Jednoduše předává volání metod z řízeného .NET kódu přímo do neřízené třídy. Řízená rozšíření mohou být použita pro vyvolání jakékoli nativní dynamicky linkované knihovny (DLL), stejně jako nativní třídy.
- Přístup k .NET třídám z nativního kódu
  Pomocí řízených rozšíření můžete zakládat a volat .NET třídy přímo z C++ kódu. Bez problému lze vytvořit C++ kód, který se chová k .NET komponentám jako každá jiná řízená C++ třída. Můžete také použít nativní podpory COM v .NET Framework a volat .NET třídy. Jestli použijete COM nebo řízená rozšíření pro přístup k .NET komponentám, bude záviset na typu projektu. V některých případech je možné zvýšit výkonnost a produktivitu vývojáře při použití řízených rozšíření.

- Řízený a nativní kód v jednom souboru
  Překladač Visual C++ převádí automaticky a transparentně data ukazatele, výjimky a tok instrukcí mezi řízeným a neřízeným kontextem. Díky tomuto procesu je možná hladká spolupráce mezi řízenými rozšířeními a neřízeným C++ kódem. Vývojářům je k dispozici velmi jemná kontrola toho, která data a kód by měla být řízená a která ne.

Schopnost výběru řízené nebo neřízené implementace každé třídy a funkce poskytuje ohromnou pružnost. Některé typy kódu nebo dat budou fungovat lépe v neřízeném prostředí. Na druhé straně, řízený kód typicky nabízí zvýšenou produktivitu vývojáře, díky vlastnostem jako jsou garbage collection a knihovny tříd. Existující neřízený C++ kód lze převést na řízený po jednotlivých částech a zachovat tak existující investice.

Vylepšení datového přístupu

ADO.NET je evolučním vylepšením Microsoft ActiveX Data Objects (ADO), které realizují spolupráci s platformou a škálovatelný přístup k datům. Libovolná aplikace, která rozumí XML může zapracovávat data, protože Extensible Markup Language (XML) je formátem pro přenos dat na platformě ADO.NET. Aplikace přijímající data může běžet na libovolné platformě.

Pomocí Visual Studia.NET můžete programovat proti objektům a ne proti tabulkám a sloupcům. ADO.NET používá silně typové programování, ve kterém významně vystupují obchodní objekty.

Uvažujme například následující řádek kódu, používající konvenční (bez striktních typů) programování:

```
IF CelkoveNaklady > Table( Zakaznik ).Column ( DostupnyKredit)
```

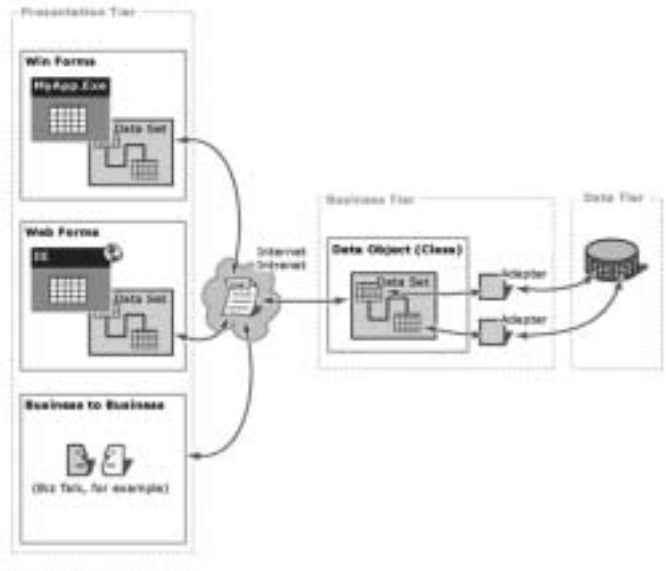
V tomto příkladu programujete ADO tabulky a sloupce. Se silně typovým programováním bude stejný příklad mnohem čitelnější a lépe programovatelný:

```
IF CelkoveNaklady > Zakaznik.DostupnyKredit
```

ADO.NET Data Sety

Základní součástí každého programového řešení používajícího ADO.NET je data set. Data set je in-memory kopií dat databáze. Data set obsahuje libovolný počet datových tabulek, kde každá z nich odpovídá databázové tabulce nebo pohledu, společně s přiřazenými relacemi a omezeními. Data set vytváří odpojený pohled na data databáze. Existuje v paměti bez aktivního připojení do databáze, která obsahuje odpovídající tabulky nebo pohledy.

Za běhu aplikace jsou data přečtena z databáze do obchodních objektů na střední vrstvě a pak zobrazena v uživatelské rozhraní. Při realizaci výměny dat používá ADO.NET perzistenci a přenosy ve formátu XML. Při přenášení dat z jedné vrstvy na druhou, řešení s ADO.NET formátují data z paměti (data set) do XML a následovně tento XML dokument posílají jiným komponentám.



Obrázek 20 Hlavní komponenty ADO.NET řešení

Existuje mnoho nových vlastností, pomocí kterých vám Visual Studio.NET výrazně usnadní práci s daty. Pro zkušené XML vývojáře je k dispozici pokročilý XML návrhář s barevným rozlišením kódu, automatickým doplňováním výrazů a tagů.

Pro grafický pohled na data mohou vývojáři použít Design View XML návrháře. Ten vám umožní přetahování tabulek z libovolného datového zdroje, včetně Microsoft SQL serveru a Oracle databází, nacházejících se na návrhové ploše záložky Server Explorer. Data sety lze vytvářet z velkého počtu zdrojů, včetně libovolného XML souboru.

Často potřebujete v průběhu návrhu aplikace přidat, modifikovat nebo mazat data. Pomocí dialogového boxu Data Preview můžete data nejen přidávat a modifikovat, ale procházet po jednotlivých relacích mezi nimi.

Technologie svázání s daty byly ve Visual Studiu.NET dramaticky vylepšeny, aby využily veškeré výhody ADO.NET. Vytvoření uživatelského rozhraní pracujícího s daty je nyní mnohem snadnější. Mnohem důležitější je, že si nyní můžete dovolit připojit hodnoty k obchodním objektům a webovým službám.

Sdílení dat s ADO.NET

Pro podporu výměny dat mezi data sety a datovými zdroji v pozadí, architektura ADO.NET používá objekty nazvané data adaptér. Microsoft dodává dva typy objektů data adaptérů:

- Objekt SqlDataAdapter: Tento objekt je prostředníkem v komunikaci mezi tabulkou data setu a tabulkou nebo pohledem v SQL Server databázi.
- Objekt OleDbDataAdapter: Tento objekt je prostředníkem v komunikaci mezi tabulkou data setu a tabulkou nebo pohledem v libovolném datovém zdroji, která je přístupný před OLE DB poskytovatele.

ADO.NET nabízí několik výhod oproti předchozí verzi ADO i oproti jiným komponentám datového přístupu. Výhody spadají do následujících kategorií:

- Interoperabilita: ADO.NET řešení mohou využít výhody pružnosti a širokého přijetí XML. Protože XML je formátem pro přenášení data setů mezi komponentami, libovolná komponenta (na libovolné platformě), která umí přečíst XML formát, může data zpracovávat. Při přenášení odpojeného ADO recordsetu z jedné komponenty na druhou, přijímací komponenta musí podporovat COM marshalling. To znamená, že musí být COM komponentou. Takové omezení se nevztahuje na přenášení ADO.NET data setu z jedné komponenty na druhou. Protože přenos data setu probíhá pomocí XML souborů a protože XML je jednoduchý textově orientovaný standard akceptovaný širokým průmyslem, na přijímací komponentu nejsou kladeny vůbec žádné architektonické nároky. V podstatě libovolné dvě komponenty mohou sdílet data sety, za předpokladu, že obě komponenty používají stejné XML schéma pro formátování přenášeného XML data setu.
- Udržovatelnost: Během životnosti nasazeného systému může dojít k mírným změnám. Pokusy o zásadní, architektonické změny jsou prováděny jen výjimečně, protože jsou velice obtížné. Je to velká škoda, neboť v průběhu přirozeného vývoje a událostí jsou takové zásadní změny zapotřebí. Například, jak se postupně stává nasazená aplikace populární uvnitř uživatelské komunity, zvýšení výkonnostního zatížení může vyžadovat architektonické změny. S nárůstem výkonnostního zatížení serveru s nasazenou aplikací může dojít k nedostatku systémových zdrojů, prodloužení reakční doby nebo omezení přenosové rychlosti. Softwarový architekti stojící před tímto problémem se mohou rozhodnout pro rozdělení zpracování serverové obchodní logiky a uživatelského rozhraní do dvou oddělených vrstev na dvou samostatných počítačích. Ve výsledku je aplikační serverová vrstva rozdělena na vrstvy dvě, které překonávají nedostatek systémových zdrojů.
  Pokud je původní řešení implementováno v ADO.NET a používá data sety, uskutečnění této transformace je velice snadné. Zapamatujte si, pokud nahradíte jednu vrstvu dvěmi vrstvami, vždy je uspořádáte tak, že si vyměňují informace. Jelikož uvedené dvě vrstvy mohou přenášet data formou XML-formátovaných data setů, komunikace je relativně jednoduchá.
- Výkonnost: ADO.NET data sety nabízí u odpojených aplikací výhodu výkonnosti před ADO odpojenými record sety. Situace, která používá COM marshalling pro přenášení odpojené množiny dat mezi vrstvami, může mít za následek výkonnostní pokles, pramenící z převádění hodnot obsažených v record setu na datové typy podporované v COM. V ADO.NET k podobné konverzi vůbec nedochází.
- Škálovatelnost: Webové služby mohou často způsobit zvýšenou poptávku po vašich datech. Škálovatelnost se proto stává obzvláště důležitým faktorem charakterizujícím kvalitu vaší aplikace. Pokud aplikace používá webové rozhraní, existuje téměř neomezená množina potenciálních uživatelů. Ačkoli aplikace může kvalitně obsluhovat desítku uživatelů, nemusí se stejnou kvalitou obsluhovat stovku. Největší překážkou škálovatelnosti je boj o omezené systémové zdroje. Pokud aplikace spotřebovává zdroje jako databázové zámky a databázová připojení, aplikace nebude schopna kvalitně obsloužit velké množství uživatelů. Jak jejich počet vzrůstá, požadavky uživatelů na zdroje může případně překročit jejich dostupné množství. Proto aplikace, která dobře obslouží desítku uživatelů, může velice chabě obsloužit stovky.

ADO.NET se vypořádává se škálovatelností tak, že nabádá programátory, aby šetřili omezenými zdroji, o které uživatelé často soupeří. Protože libovolná ADO.NET aplikace používá odpojený přístup k datům databáze, nedrží po delší dobu databázové zámky nebo aktivní databázová připojení. Sníží se tak využívání omezených databázových zdrojů.

Přenášení ADO.NET data setu mezi vrstvami nebo komponentami je jednodušší než přenášení odpojeného ADO record setu. Pro přenesení odpojeného ADO record setu z jedné komponenty na druhou musíte použít COM marshalling. Pro přenesení ADO.NET data setu jednoduše přenesete XML soubor. Přenos XML souboru nabízí následující výhody před COM marshalling:

- Bohaté datové typy: COM marshalling podporuje omezenou množinu datových typů – datové typy definované COM standardem. Protože přenos data setu je založen na formátu XML, není zde žádné omezení vztahující se na datové typy. Komponenty sdílející data set mohou pracovat s naprosto libovolnou množinou datových typů, kterou by běžně používali.
- Výkonnost: Přenesení velkého ADO record setu nebo velkého ADO.NET data setu může spotřebovat síťové zdroje. Jak roste objem dat, vzrůstá i tlak na síť. Jak ADO, tak ADO.NET mohou minimalizovat množství přenášených dat. ADO.NET však nabízí další výkonnostní výhodu v tom ohledu, že nevyžaduje žádné typové konverze. Naproti tomu ADO potřebuje COM marshalling k přenesení record setu mezi komponentami, zahrnující konverzi mezi datovými typy ADO a COM.
- Průchod přes firewally: Firewall může překážet operaci, při které se dvě komponenty snaží přenést odpojený ADO record set. Firewally jsou typicky nakonfigurovány tak, aby propustily HTML text, ale zabránily průchodu dotazům na systémové úrovni (jako je COM marshalling). Při výměně ADO.NET data setů pomocí XML, firewally je bez problémů propustí.

Odpojený přístup k databázím

ADO.NET data set poskytuje odpojený přístup k datům z databáze. Record set v ADO mohl poskytnout odpojený přístup, ale typicky byl používán v připojeném režimu.

Mezi odpojeným zpracováním v ADO a ADO.NET existuje jeden významný rozdíl. V ADO komunikujete s databází formou volání služeb OLE DB poskytovatele. V ADO.NET však komunikujte s databází prostřednictvím objektu DataAdapter, který interně volá OLE DB poskytovatele (v některých případech používá přímé API rozhraní poskytované Systémem řízení databáze (DBMS)).

Důležitým rozdílem je, že můžete modifikovat kód objektu DataAdapter. Ve výsledném efektu můžete řídit, jak jsou změny v data setu přenášeny zpět do databáze. Znamená to, že můžete provést výkonnostní optimalizaci, vykonávat validaci dat nebo provádět libovolné dodatečné a speciální zpracování dat.

PŘÍLOHA

Odkazy na zdroje informací

Následující seznam není úplný a vyčerpávající seznam článků, ale nasměruje vás na nejsrozumitelnější články o .NET.

Technologie
XML:
http://msdn.microsoft.com/xml/default.asp

Informace o COM:
http://www.microsoft.com/net/developer/framework_com.asp

Vývoj Microsoft DNA:
http://www.microsoft.com/business/products/webplatform/default.asp

Informace o UDDI:
http://www.uddi.org
http://www.microsoft.com/presspass/features/2000/sept00/09-06uddi.asp

Webové služby a .NET platforma:
http://msdn.microsoft.com/msdnmag/Issues/0900/Framework/print.asp
http://msdn.microsoft.com/msdnmag/Issues/1000/Framework2/print.asp
http://msdn.microsoft.com/msdnmag/Issues/0900/WebPlatform/WebPlatform.asp
http://msdn.microsoft.com/msdnmag/Issues/0800/webservice/print.asp
http://msdn.microsoft.com/msdnmag/Issues/0300/soap/print.asp
http://www.microsoft.com/net/xmlservices.asp

Garbage Collection:
http://msdn.microsoft.com/msdnmag/Issues/1100/gcl/print.asp

Nasazení a podpora uživatelů
http://msdn.microsoft.com/msdnmag/Issues/1000/metadata/print.asp

Vývojáři
Technologie a zdroje pro vývojáře:
http://msdn.microsoft.com/resources

Technologie a architektura .NET
http://www.microsoft.com/net/
http://msdn.microsoft.com/net/

MSDN magazín:
http://msdn.microsoft.com/msdnmag

Webové služby:
http://msdn.microsoft.com/msdnmag/Issues/0900/VSNET/print.asp
http://msdn.microsoft.com/msdnmag/Issues/0900/Webplatform/print.asp
http://msdn.microsoft.com/msdnmag/Issues/0800/Webservice/print.asp

Zrychlený vývoj serverových aplikací:
http://msdn.microsoft.com/vstudio/nextgen/technology/radserver.asp

ASP.NET:
http://msdn.microsoft.com/msdnmag/Issues/0900/aspplus/print.asp

Jazyky:
http://msdn.microsoft.com/library/welcome/dsmsdn/deep07202000.htm

Obecné stránky
http://www.microsoft.com/net
http://msdn.microsoft.com/net
http://www.GotDotNet.com
http://www.asp.net
http://www.microsoft.com/servers/net/default.htm
http://www.ibuyspy.com

Kapitola 4:

XML, webové služby a .NET Framework

ÚVOD

Vývoj aplikací prochází významnou změnou – změnou, která jednoznačně zvýší produktivitu vývojářů a otevře cesty k novým typům aplikací.

Z historického pohledu stavěli vývojáři aplikace formou integrace lokálních systémových služeb. Tento model jim poskytnul přístup k široké množině vývojových zdrojů a přesnou kontrolu nad tím, jak se aplikace chovají.

Vývojáři se posunuli již mnohem dále za tento model. V současnosti budují složité N-vrstvé systémy, které integrují dohromady celé aplikace, nacházející se ve všemožných sítích a přidávají jim vlastní užitnou hodnotu. Vývojáři se mohou koncentrovat na jedinečnou obchodní hodnotu místo budování infrastruktury. Výsledkem je kratší doba uvedení na trh, vyšší vývojová produktivita a bezprecedentně kvalitnější software.

Stáváme se účastníky dalšího posunu, který je způsoben přítomností Internetu - konkrétně klíčovou Internetovou technologií XML. Zatímco vývoj N-vrstvých aplikací se zaměřil na vytváření velkých aplikací pro firemní použití, XML umožňuje vytváření velkých aplikací, které mohou být používány kýmkoli a kdekoli. Zároveň zvyšuje dosah aplikací a umožňuje průběžnou dodávku programového vybavení, nikoli ve stylu instalace z CD, ale jako službu – službu ve stylu platba-za-přístup.

Realizace je provedena spojením úzce vázaných, vysoce produktivních aspektů N-vrstvých výpočtů s volně vázanými, na zprávách postavených koncepcích Webu. Tento styl výpočetních systémů je nazván webová služba a představuje další evoluci vývoje aplikací. Webová služba je aplikace, která programově zveřejňuje své schopnosti na Internetu nebo intranetu za použití standardních internetových protokolů jako jsou HTTP a XML. Uvažujte o ní jako o komponentovém programování na Webu.

Z koncepčního pohledu vývojáři integrují webové služby do svých aplikací voláním „Web API“ stejně, jako by volali lokální služby. Rozdíl spočívá v tom, že tato volání mohou být směrována přes Internet ke službám sídlícím na vzdálených systémech. Například služba jako Microsoft Passport může vývojářům poskytnout podporu autentikace v aplikacích. Programováním proti službě Passportu vývojář využije výhody infrastruktury, na níž je Passport postaven a spoléhá se na údržbu databáze uživatelů, zajištění spolehlivého běhu, korektní zálohování dat, atd.

Volné svázání

Koncepce vytváření distribuované aplikační logiky po síti není novinkou. Koncepce distribuované a integrované aplikační logiky na Webu však je.

Distribuovaná aplikační logika v minulosti využívala distribuovaný objektový model jako je Microsoft DCOM, CORBA skupiny OMG nebo RMI firmy Sun. Vývojáři si mohou díky této infrastruktuře udržet mnoho z bohatství a preciznosti, na kterou byli zvyklí z lokálního programového modelu a být stále schopni umístit služby na vzdálené systémy.

Problémem těchto systémů je, že neškálují v Internetu: jejich závislost na úzké vazbě mezi konzumentem služby a službou vlastní si vynucuje homogenní infrastrukturu a často znamená, že jsou velice křehké. Pokud se implementace na jednom z konců změní, druhá strana přestane fungovat. Pokud se například změní rozhranní serverové aplikace, klient pak snadno havaruje.

Ve skutečnosti není nic špatného na požadavku úzce vázané infrastruktury a mnoho aplikací bylo postaveno s její pomocí. Tento model však bez diskuse nebude škálovat v čase. Jak se jednotlivé společnosti navzájem propojují nebo když firmy z oblasti informačních technologií odcházejí ze scény, je velice obtížné garantovat jednu unifikovanou infrastrukturu. Neexistuje žádná záruka, že služba, se kterou chcete komunikovat na vzdáleném konci drátu, bude mít veškerou infrastrukturu, kterou požadujete. Nemusíte ani vědět, jaký operační systém, objektový model nebo programovací jazyk používá.

Webové služby jsou volně vázané – můžete změnit implementaci na libovolném konci spojeni a aplikace budou stále fungovat. Technicky lze tuto vlastnost realizovat pomocí zpráv, asynchronních technologií zajišťujících robustnost, použitím webových standardních protokolů jako HTTP, SMTP a ze všeho nejdůležitěji XML, s jehož pomocí dosáhneme univerzálního rozšíření.

Messaging systémy zaobalují základní jednotky komunikace do samopopisných jednotek (nazvané zprávy – message) přenášených síťovým médiem. Klíčový rozdíl mezi messaging systémem a distribuovaným objektovým systémem spočívá v tom, kolik znalostí o infrastruktuře příjemce vyžaduje odesilatel. U distribuovaného objektového systému pracuje odesilatel s předpoklady o příjemci – o tom, jak bude aplikace aktivována a jak ukončena, jaká rozhranní budou volána atd.

Messaging systémy na druhé straně vytváří kontrakt na úrovni přenosového formátu. Jediný předpoklad, která odesilatel má, se vztahuje k otázce, zda příjemce porozumí odeslané zprávě. Odesilatel nevytváří žádné předpoklady o tom, co se bude dít, jakmile je zpráva přijata a ani o tom, co by se mohlo stát mezi odesilatel a příjemcem.

Výhoda plynoucí z vytvoření kontraktu na úrovni přenosového formátu je docela zřejmá. Umožní například změnit příjemce v libovolné chvíli, aniž by porušila odesilatele, za předpokladu, že budou nadále rozumět stejným zprávám. Odesilatel dokonce nevyžaduje ani žádný speciální software, aby byl schopen komunikovat s příjemcem. To vše za podmínek, že vysílá korektně formátované zprávy, na které příjemce odpovídá.

XML po drátě: SOAP

Klíčem ke zpřístupnění webových služeb napříč Webem a jeho heterogenní infrastruktuře je dohodnutí jednoduchého formátu popisu dat. Tímto formátem je XML. Webové služby potřebují XML konkrétně ke třem věcem: základní přenosový formát, popis služeb a vyhledávání služeb.

- **SOAP.** Na nejnižší úrovni potřebují systémy komunikovat společným jazykem. Konkrétně, komunikující aplikace musí mít sadu pravidel určujících, jak budou reprezentovat různé datové typy (např. celá čísla nebo pole) a jak budou reprezentovány příkazy (tzn. co má být s daty provedeno). Aplikace také potřebují způsob, jakým rozšířit tento jazyk, pokud je to nutné. Simple Object Access Protocol (SOAP), implementace XML, představuje jednu společnou množinu pravidel určující, jak budou data a příkazy reprezentovány a rozšiřovány.
- **Web Service Description Language.** Jakmile mají aplikace obecná pravidla jakými budou reprezentovány datové typy a příkazy, potřebují najít způsob popisu specifických dat a příkazů, které jsou schopny přijmout. Nestačí, aby aplikace prohlásila, že přijímá celá čísla. Někde musí existovat deterministický způsob, jakým prohlásit, že dvě předaná celá čísla mají být navzájem vynásobena. Web Service Description Language je XML gramatika kterou mohou vývojáři a vývojové nástroje použít k představení vlastností webové služby.
- **SOAP Discovery.** Poslední vrstva, kterou potřebujete, slouží k nastavení pravidel týkajících se nalezení popisu služby – kam se má člověk nebo nástroj implicitně podívat, aby našel popis možností služby? Specifikace SOAP Discovery poskytuje sadu pravidel sloužících vývojovým nástrojům nebo lidem při automatickém vyhledání popisu služby v jazyce WSDL.

Jakmile existují uvedené tři vrstvy, vývojáři mohou webové služby jednoduše nalézt, vytvořit je jako objekty, integrovat je do svých aplikací a vytvořit dostatek infrastruktury proto, aby s nimi mohly aplikace komunikovat.

.NET Framework: Jádro webových služeb

Je zřejmé, že potřebujeme nezanedbatelné množství infrastruktury, aby tento proces byl pro vývojáře i uživatele zcela transparentní. .NET Framework ji poskytuje. Uvnitř .NET Framework mohou být všechny komponenty webovými službami a webová služba je jen dalším typem komponenty. Ve skutečnosti .NET Framework vybírá ty nejlepší aspekty COM (Microsoft Component Object Model) a kombinuje je s nejlepšími aspekty volně vázaných výpočtů. Výsledkem je výkonný a produktivní webový komponentový systém, který zjednodušuje infrastrukturní kód, hluboce integruje bezpečnost, zavádí internetově škálovatelný systém nasazení a významně vylepšuje spolehlivost a škálovatelnost aplikací.

.NET Framework se skládá ze tří hlavních částí: Common Language Runtime, hierarchická množina jednotných knihoven tříd a pokročilé verze Active Server Pages nazvaná ASP.NET.

Common Language Runtime

Runtime, navzdory svému jménu, sehrává svou roli ve skutečnosti při vývoji a běhu komponent. Za běhu komponenty je runtime zodpovědný za management paměti, startování k ukončování threadů, vynucení bezpečnostní politiky, stejně jako vyhovění závislostem, které komponenty mohou mít na ostatních komponentách. V době vývoje se role runtime poněkud mění. Diky značné automatizaci (např. řízení paměti), runtime extrémně ulehčuje život vývojářům, zejména při srovnání se současným COM. Konkrétně, vlastnosti jako jsou reflexe dramaticky redukují množství kódu, které vývojář musí napsat k tomu, aby převedl obchodní logiku na opakovatelně použitelné komponenty.

Runtime není v programovacích jazycích žádnou novinkou. Prakticky všechny programovací jazyky mají runtime. Visual Basic je tím nejzřetelnějším (vhodně pojmenovaný VBRUN), Visual C++ má svůj (MSVCRT), ale i Visual FoxPro, JScript, SmallTalk, Perl, Python a Java. Kritická úloha .NET Frameworku, která jej ve skutečnosti odlišuje, spočívá v existenci jednotného prostředí společného pro všechny programovací jazyky.

Jednotné programové třídy

.NET Framework třídy poskytují vývojářům unifikovanou objektově orientovanou hierarchickou a rozšiřitelnou množinu knihoven tříd („API“). C++ vývojáři v současnosti používají Microsoft Foundation Classes, Java vývojáři Windows Foundation Classes a programátoři v jazyce Visual Basic používají VB API. Jednoduše řečeno, .NET Framework sjednocuje roztříštěné nástroje, které Microsoft v současné době poskytuje. Výsledkem je, že se již vývojáři nemusí nadále učit několik různých prostředí a API. Vytvořením společného API napříč všemi programovacími jazyky umožňuje .NET Framework dědičnost, zpracování chyb a ladění bez jazykových hranic. Ve výsledném efektu se všechny programovací jazyky, počínaje JScript až po C++, stávají rovnými a vývojáři si mohou svobodně vybrat jazyk, který jim vyhovuje.

Active Server Pages.NET

ASP.NET je postaveno nad .NET Framework třídami. Realizuje „webový programový model“ ve formátu množiny ovládacích prvků a infrastruktury, zjednodušující vývoj webových aplikací. Vývojářům je dostupná množina ovládacích prvků, které zaobalují HTML prvky běžného uživatelského rozhranní jako jsou text boxy, spustitelná menu a podobně. Tyto prvky však ve skutečnosti běží na webovém serveru a jednoduše promítají své uživatelské rozhranní prohlížeči jako HTML. Ovládací prvek na serveru vystavuje objektově orientovaný model, který přináší webovým vývojářům nádheru objektově orientovaného programování. ASP.NET také poskytuje infrastrukturní služby, například řízení stavu relace a recyklaci procesu, které dále snižují množství kódu, jenž by vývojář musel napsat pro zvýšení spolehlivosti aplikací. ASP.NET používá stejných koncepcí, aby vývojářům umožnil dodávat software ve formátu služeb. ASP.NET vývojář může, při využití schopností ASP.NET webových služeb, velice jednoduše psát aplikační logiku a infrastruktura ASP.NET se postará o zpřístupnění služeb přes protokol SOAP.

Ačkoli je ASP.NET velice krátce na světě, již od počátku vykazuje některá ohromující vylepšení v oblasti webových aplikací – zvýšení výkonnosti více než 3X oproti existujícím aplikacím napsaným v ASP a zvýšení produktivity ještě mnohem drastičtějším způsobem.

Klíčové aspekty .NET Framework

.NET Framework disponuje několika aspekty, které se stanou populárními. Mezi nejdůležitější patří systémy nasazení a bezpečnosti. Tyto dva systému vzájemně spolupracují za účelem snížení pravděpodobnosti spuštění nebezpečného kódu a omezení některých problémů s nasazením aplikací, často popisovaných jako „DLL Hell“.

Bezpečnostní systém je jemně členěný a založený na faktech (evidence). Co to znamená? Znamená to, že vývojářům a administrátorům nabízí široký rozsah privilegií, která mohou přiřadit kódu (nikoli jen „zakázat“ a „povolit“) a jak tato privilegia budou aplikována, záleží na klíčových aspektech kódu samotného.

Když je .NET Framework aplikace zaváděna do systému, bude žádat o jistou množinu povolení – například o povolení zápisu do odkládacího adresáře. Runtime zjistí fakta (evidence) o aplikaci, jako odkud byla nahrána, jestli je podepsána pomocí Authenticode podpisu, dokonce k čemu přesně se snaží v systému přistoupit a začne spolupracovat s administrativně nastavenou množinou politik. Tak zjistí, zda by se aplikace měla spustit nebo ne. Runtime může dokonce aplikaci oznámit, že není schopen garantovat všechna požadovaná povolení a dát aplikaci na výběr, jestli chce pokračovat v běhu.

S takto připraveným bezpečnostním systémem je významně zjednodušeno řešení problému nasazení aplikace. Jedním z největších úskalí, kterému čelí vývojáři a administrátoři (a bezpochyby uživatelé) je správa verzí. Pokud váš systém funguje jeden den, následně instalujete novou aplikaci a najednou přestane fungovat, je velmi často důvodem přepsání některých sdílených knihoven dodaných společně s novou aplikací. Velmi často jde o opravy chyb, ale i změnu jistého chování, na které se existující aplikace spoléhají. Jde o tak častou věc, že dokonce dostala své jméno: DLL Hell.

.NET Framework obsahuje vylepšení, která prakticky odstraňují tento problém. Zaprvé obsahuje velmi silný interní jmenný systém, poskytující odlišné jmenné prostory. Znamená to, že neexistuje žádný způsob, jak si splést dvě knihovny, i když nesou stejná fyzická jména souborů. Existuje také zcela nová vlastnost nazvaná „souběžné nasazení“ (side-by-side deployment). Pokud aplikace z předchozího příkladu skutečně přepíše sdílenou knihovnu, existující aplikace je schopna vlastní opravy. Ve chvíli, kdy je starší aplikace znovu spuštěna, dochází ke kontrole všech sdílených souborů. Zjistí-li se modifikace v jednom z nich a tyto modifikace jsou nekompatibilní, požádá CLR, aby dodalo verzi o niž ví, že je kompatibilní. Díky systému bezpečnosti a řízení verzí, CLR spolehlivě zvládne tuto operaci a aplikace se může sama opravit.

ZÁVĚR

Příliš často lidé používají otřepané fráze jako „Internet mění vše“. Příliš často hovoří ve jménu marketingové nadsázky. Internet však skutečně změnil základním způsobem vývoj aplikací a jejich nasazení. Poskytování programového vybavení jako služeb je budoucností a XML je základem k jeho realizaci. .NET Framework je klíčovou součástí vývojové strategie Microsoftu, směřující k ulehčení vývoje, nasazení a běhu webových služeb.

Na okraj: Vztah k technologiím COM a COM+

Jedním z primárních cílů .NET Framework je zjednodušit vývoj COM aplikací. Pravděpodobně nejtěžší věcí, týkající se COM vývoje, byla práce s COM infrastrukturou. Aby se vývoj v oblasti COM usnadnil, .NET Framework automatizuje mnoho obtížné práce, o které vývojáři přemýšlejí jako o „COMu“. Jde zejména o počítání referencí, popis rozhranní a registraci.

Je důležité zmínit, že to neznamená, že by .NET komponenty nebyly COM komponentami. COM vývojáři pracující s Visual Studiem 6.0 mohou ve skutečnosti volat .NET Framework komponenty a bude jim připadat jako by pracovali se skutečnými COM komponenty, včetně rozhranní IUnknown. A naopak, .NET Framework vývojáři, pracující s Visual Studiem.NET, mohou pohlížet na COM komponenty, jako by šlo o .NET Framework komponenty.

COM+ je název pro kombinaci COM komponent s Microsoft Transaction Serverem, neboli MTS a technologií DCOM. COM+ poskytuje sadu služeb orientovaných na střední vrstvu. COM+ konkrétně realizuje management procesů, pooling objektů a databázových spojení. V budoucích verzích bude také poskytovat silnější izolaci procesů navrženou pro poskytovatele aplikačních služeb – vlastnost nazvanou „partitioning“.

COM+ služby jsou primárně orientovány na vývoj střední vrstvy a zaměřeny na zvýšení spolehlivosti a škálovatelnosti pro velmi rozsáhlé distribuované aplikace. Tyto služby jsou doplňkem k programovým službám dostupným v .NET Framework. .NET Framework třídy poskytují přímý přístup k popsaným komponentově orientovaným službám.

Na COM+ můžete pohlížet jako na zdroj komponentových služeb pro vytváření škálovatelných, spolehlivě distribuovatelných aplikací střední vrstvy. .NET Framework nabízí jazykové služby, které zjednodušují a urychlují vývoj a nasazení aplikací.

Kapitola 5:

Úvod do webových služeb: WSDL, SOAP a UDDI

ZÁKLADY WEBOVÝCH SLUŽEB

V posledních deseti letech nemělo nic větší vliv na svět software než Internet a World Wide Web. Aplikace postavené na prohlížečích se staly postupně standardem, jak se organizace začaly přizpůsobovat novému prostředí. I přes svou užitečnost však nejsou aplikace, zveřejňující své služby prohlížečům, jediným způsobem, jak využít technologii Webu. Proč nemohou tyto aplikace také zveřejnit své služby programovým způsobem? Umožníme-li programovému vybavení běžícímu na jiných systémech přímo vyvolávat operace přes Web, dovolíme webovým aplikacím nabízet bohatší služby mnoha odlišným klientům.

Díky webovým službám se myšlenka stávají skutečností. S novou technologií může software, běžící na stolních počítačích, osobních digitálních asistentech (PDA), mobilních telefonech nebo na libovolném inteligentním zařízení, přímo vyvolávat operace zveřejněné webovými aplikacemi. Zatímco aplikace dostupné v Internetu budou pravděpodobně tvořit velkou část nového světa, mohou aplikace běžící na interních sítích také využívat výhod pramenících z webových služeb. Cílem kapitoly je poskytnout krátký úvod ke klíčovým technologiím stojícím v pozadí této nové myšlenky.

Co je webová služba?

Technologie webových služeb lze rozdělit do čtyř samostatných oblastí, dotýkajících se jednotlivých konkrétních aspektů problému. Jsou to:

- **Popsání informací zasílaných po síti.** Vyvolání vzdálené operace obecně zahrnuje předání parametrů a získání jistého typu výsledku. U webových služeb jsou informace popsány prostřednictvím Extensible Markup Language (XML). XML, všeobecně schválená moderní ligua franca pro popis dat, umožňuje charakterizovat a přenášet všechny typy informací.
- **Definování schopností webové služby.** Musí existovat jisté mechanismy sloužící poskytovateli webové služby ke specifikaci technických podrobností, které konkrétní webová služba nabízí. Obdobně jako u jiných typů služeb, dává smysl seskupit příbuzné operace do rozhranní a následovně poskytnout jistý způsob, jak popsat každou z těchto operací. U webových služeb můžeme použít Web Service Description Language (WSDL – v anglicky mluvících zemích vyslovováno jako „wizdl“). Každé rozhranní definované pomocí WSDL obsahuje jednu nebo více operací a každá operace má výstupy a vstupy definované v XML.
- **Přistoupení k webové službě.** Jakmile je rozhranní definováno, klienti musí použít nějaký protokol k vyvolání operací v rozhranní. Neexistuje pouze jeden protokol (ve skutečnosti WSDL explicitně určuje různé typy protokolů pro vyvolání operací rozhranní). Pravděpodobně nejdůležitější volbou je Simple Object Access Protocol (SOAP). SOAP je definován pomocí XML a poskytuje způsob, jak určit, která operace má být vyvolána, předat této operaci vstupní data ve formátu XML a vrátit libovolný výstup, opět jako XML data. SOAP sám o sobě definuje pouze jednoduchou obálku pro dopravení informací, obálku, kterou lze přenášet různými způsoby – například jako HyperText Transfer Protocol (HTTP) nebo Simple Message Transfer Protocol (SMTP).
- **Nalezení Webové služby.** Aby mohli vývojáři vytvářet klientské aplikace, které používají webové služby, musí existovat jistý způsob, jak se dozví, jaké služby jsou k dispozici, co poskytují a jak vypadá jejich rozhranní. Vezmeme-li v úvahu existenci Internetu, nabízí se myšlenka vytvoření standardního registru pro ukládání a zveřejnění takových informací. Je to přesně to, k čemu slouží technologie definovaná v Universal Description, Discovery, and Integration (UDDI). Poskytovatelé webových služeb mohou standardním způsobem pomocí UDDI propagovat své nabídky a umožnit tvůrcům klientského software zjistit cokoli, co potřebují k jeho vytvoření.

Každá s těchto technologií byla vytvořena skupinou spolupracujících výrobců a uživatelů. XML byl například vytvořen velkou skupinou pod záštitou World Wide Web konsorciem (W3C), zatímco WSDL byl vytvořen firmami Microsoft a IBM. SOAP pochází od skupiny střední velikosti, začínající firmami DevelopMentor a Microsoft a později sehrála svou roli i IBM a několik dalších organizací. UDDI, nejnovější z uvedených technologií, byla původně vytvořena firmami Microsoft, IBM a Ariba, i když se od té doby připojilo k iniciativě mnoho dalších organizací.

Klíčovým bodem, týkajícím se původu každé technologie webových služeb je, že nejde o řešení postavené pouze na jednom výrobci. Ve skutečnosti žádná z nich není svázána s proprietárními technologiemi jako Windows, Unix, Component Object Model (COM) nebo Enterprise JavaBeans (EJB). Webové služby založené na WSDL, SOAP a UDDI mohou být místo toho použity prakticky na všech platformách, ve všech jazycích a na každém objektovém modelu.

Použití webových služeb

Existuje mnoho možných scénářů pro nasazení webových služeb. Snad tím nejjasnějším je představa, že koncoví uživatelé budou moci vyvolávat operace poskytované aplikacemi přístupnými po Internetu. Přestavte si například, že by jste si chtěli objednat návštěvu u svého zubaře, objednat si let do Paříže nebo zjistit zůstatek na svém bankovním účtu. Všechny tyto věci můžete dnes provést pomocí klasických Web aplikací. Aplikace poskytují své služby pomocí grafického uživatelského rozhraní (GUI), avšak vyžadují docela pěkný kus interakce s člověkem. Díky své přílišné grafice jsou mnohem méně přitažlivé, pokud je klientem zařízení do ruky typu PocketPC nebo mobilní telefon. Všechny tyto funkce a mnoho dalších by mohlo být vyřešeno mnohem efektivněji, pokud by byly dostupné jako webové služby. Při použití jednoduchého GUI, odpovídajícího danému typu klientského zařízení, by jste mohli vyjádřit svá přání a následovně je nechat zpracovat webovými službami. Zatímco aplikace využívající prohlížeče udělaly z Webu to, čím dnes je, je zcela možné, že webové služby z něj udělají to, čím bude zítra.

Použití služeb tímto způsobem vytváří také další příležitost: proč by se nedal pomocí služeb vytvářet zisk? Stejně jako Web aplikace s GUI rozhraním umožnily nový typ obchodního modelu, který vyvolal Internetový boom, Web služby mohou také připravit nové způsoby, jak vydělávat peníze. Mnoho webových služeb může například požadovat stejné informace o svých uživatelích, jako jsou autentikační data nebo poštovní adresa uživatele. Místo toho, abychom žádali jejich individuální zadání po každém uživateli, obecná webová služba je může centrálně uchovat a za poplatek poskytovat přístup mnoha různým klientům. Microsoft je jednou z organizací, která věří v tento obchodní model a ohlásila své plány poskytovat různé standardní stavební bloky služeb v prostředí Internetu.

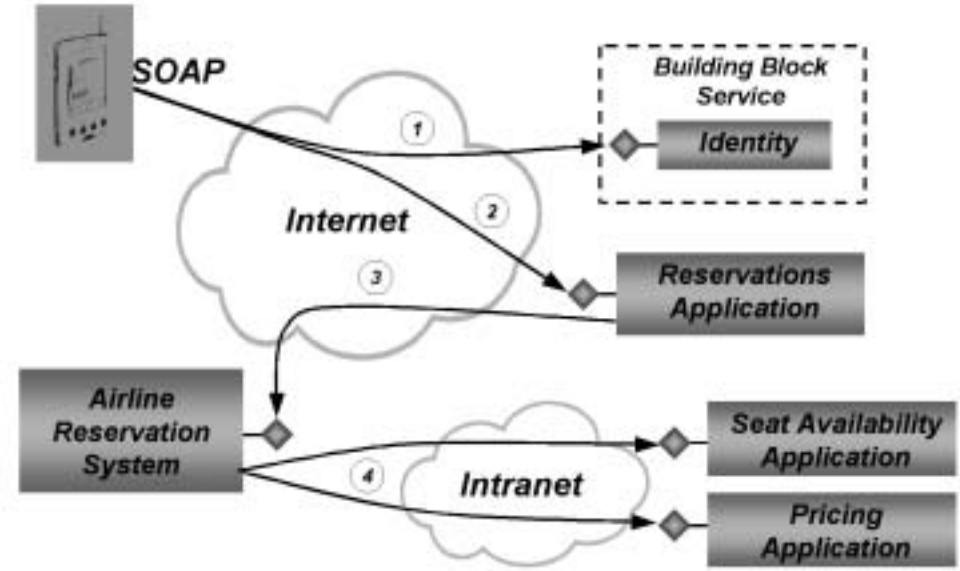
Další důležitou aplikací webových služeb, která je také závislá na Internetu, je business-to-business (B2B) integrace mezi rozmanitými organizacemi. Propojení aplikací mezi dvěmi odlišnými organizacemi bylo v současnosti často realizováno dosti zvláštními způsoby. Existují sice jisté standardy, jako Electronic Data Interchange (EDI), ale tyto formáty mají tendenci být velmi složité a výstřední. Proč nepoužít něco jednoduššího?

Webové služby jsou otevřenou, široce dostupnou technologií, která může být použita k vyřešení tohoto problému. Popis dat pomocí XML, definování rozhranní pomocí WSDL, vyhledání dostupných rozhranní s UDDI a vyvolání služeb přes SOAP může představovat společný tmel napříč všemi typy rozdílných prostředí. Další výhodou přístupu je, že webové služby mohou být eventuálně přístupné i přes firewally, protože SOAP běží nad HTTP. Přístup samozřejmě vzbuzuje řadu bezpečnostních otázek, a tak jim je věnována pozornost dále v této kapitole. Bez jistého způsobu tunelování přes firewall by však B2B integrace v Internetu nebyla možná.

Naposled zmiňovaný příklad užitečných webových služeb může být pro jisté organizace tím nejdůležitějším: integrace mezi aplikacemi uvnitř společnosti, problém pod společným názvem Enterprise Application Integration (EIA). Každá společnost libovolné velikosti vlastní směsici softwaru běžícího na různých systémech, které byly napsány v různých dobách a v různých jazycích. Spojení takto obrovského množství aplikací do jednoho užitečného celku je jednou z největších výzev, které musí společnosti čelit, zejména ve chvíli, kdy doplňují další nové aplikace. Web služby nabízejí řešení tohoto znepokojivého problému.

Aby jste pochopili jak, vzpomeňte si, že základním myšlenkou, stojící v pozadí webových služeb, je definice rozhranní, jehož operace mohou být teoreticky vyvolány libovolným klientem. Zatímco nové aplikace mohou jednoduše podporovat rozhranní webových služeb, je také možné zmodernizovat existující aplikace a vybavit je rozhranními. Protože základní technologie webových služeb nejsou vázány na konkrétní prostředí a jsou podporovány mnoha výrobci, jsou excelentní volbou pro propojování různorodých aplikací uvnitř organizace. SOAP a jemu podobné protokoly pravděpodobně uvidíme jako standardní mechanismy, které budou aplikace používat pro zveřejnění svých služeb okolnímu světu.

Existuje samozřejmě mnoho dalších možností použití webových služeb, avšak v této chvíli se zdá, že uvedené tři jsou nejdůležitější. Abychom si lépe uvědomili, jak mohou být použity, představme si, že by jste si přáli provést rezervaci letenek přímo z vašeho PocketPC pomocí bezdrátového Internetového připojení. Jakmile jste zadali cíl letu a datum, o které máte zájem, následující obrázek ilustruje, jak by systém mohl provést rezervaci cesty.



Prvním úkolem, krok 1 v diagramu, je zjištění obecně potřebných informací o klientovi ze standardní služby Identity, dostupné v Internetu. Službou může být jedna ze stavebních Microsoft služeb, jak je naznačeno v diagramu nebo i nějaká jiná. Tato služba, stejně jako všechny ostatní webové služby demonstrované v příkladu, je přístupná přes SOAP.

Následovně PocketPC kontaktuje přes Internet rezervační aplikaci, jak vidíme v kroku 2. Aplikaci může provozovat internetová cestovní společnost, proto může ve skutečnosti obsahovat pouze obecný přístupový bod k vlastnímu rezervačnímu systému poskytovaného různými leteckými společnostmi. Krok 3 diagramu ukazuje, za uvedeného předpokladu, aplikaci kontaktující konkrétní rezervační systém provozovaný konkrétní leteckou společností. Důležitou skutečností je, že se vše odehrává v Internetu, což znamená, že zmíněná letecká společnost musela zpřístupnit svůj rezervační systém Internetové cestovní službě (bez nutnosti zveřejnění komukoli jinému) jako webovou službu. Na závěr, jak vidíme v kroku 4, rezervační systém letecké společnosti provede vlastní rezervaci. Aplikace jsou dostupné pouze v intranetu společnosti a mohou tedy dobře běžet na jiném typu systému používajícím odlišné implementační technologie. Dokonce i zde webové služby sehrávají úlohu tmelu, který spojuje zcela odlišné programové vybavení.

Kroky 1 a 2 diagramu jsou příkladem přímého klientského přístupu ke službám přes Internet, tedy prvním výše uvedeným příkladem nasazení webových služeb. Krok 3 ukazuje Internetovou B2B integraci pomocí webových služeb, zatímco krok 4 ilustruje scénář EAI, který využívá webových služeb pro integraci domácích aplikací v prostředí intranetu. V každém případě tyto užitečné technologie poskytují společné rozhraní a společný způsob, jak komunikovat mezi různými aplikacemi na různých systémech a po různých typech sítí.

POPISUJEME INFORMACE: XML

Komunikace by nebyla možná bez dohody stanovující, jak reprezentovat to, co se komunikací přenáší. Během několika minulých let se objevilo XML jako standardní řešení pro popis informací přenášených mezi heterogenními systémy. Není překvapením, že se XML stalo základem webových služeb. Každá druhá technologie v rodině webových služeb jej používá jistým způsobem, a tak pochopení webových služeb vyžaduje porozumění XML. K pochopení základů webových služeb nám naštěstí stačí znát pouze několik základních koncepcí XML. Těmito klíčovými myšlenkami jsou tagy, elementy, schémata a jmenné prostory.

Tag je typicky slovo uzavřené ve špičatých závorkách, jako je `<Ucet>` nebo `<Zustatek>`. Tagy jsou používány v párech – počáteční a ukončovací element. Ukončovací tag elementu je pouhým počátečním tagem s opačným lomítkem na počátku slova. Každý element obsahuje text, který má určitý vztah ke slovu v tagu. Například, v elementu

```
<Ucet>729-1269-4785</Ucet>
```

text mezi dvěmi tagy pravděpodobně představuje číslo nějakého účtu, jako je běžný účet v bance. Obdobě v elementu

```
<Zustatek>3,822.55</Zustatek>
```

text mezi tagy bude představovat objem peněz dostupných v současné chvíli na jistém účtu. Se správnou množinou tagů je možné vytvořit XML dokument obsahující elementy, které popisují informace velice užitečným způsobem. Můžete například vytvořit jednoduchý dokument obsahující informace o bankovních účtech a zůstatcích, za použití pouze dvou právě uvedených elementů.

Jak jsou však definovány elementy samotné? Musí existovat způsob, který specifikuje skupinu elementů (a tedy i tagů), které budou používány pro konkrétní účel. Na počátku vývojové etapy XML byly elementy definovány v Document Type Definition (DTD). DTD trpěly mnoha problémy, a tak je v současné době používána mnohem pružnější metoda nazvaná XML Schéma. Každé schéma definuje jeden nebo více elementů, společně s pravidly, jak mají být tyto elementy používány. Schéma může například definovat element Obálka ohraničený tagy `<Obalka>` a `</Obalka>`, následovně povoluje, že tento element může interně obsahovat jeden nebo více výskytů elementu Tělo, označeného tagy `<Telo>` a `</Telo>`.

Pokud by jeden dokument používal tagy pouze z jednoho schématu, život by byl jednoduchý. Je však běžné kombinovat v jednom dokumentu elementy ze dvou nebo více schémat. Jelikož schémata byla vytvořena nezávisle, je zcela možné, že jsou stejné tagy definovány v obou schématech s odlišným významem. Například tag `<Ucet>` v jednom schématu může znamenat běžný účet, zatímco identický tag definovaný v druhém schématu může identifikovat element, představující hodnotu nezplacené půjčky. Abychom mohli kombinovat tagy z různých schémat v jednom dokumentu, musí existovat způsob, jak spojit element se schématem, v němž je definován. Zprostředkování této asociace je úlohou jmenných prostorů.

Jmenný prostor je globálně jedinečné jméno pro množinu elementů. Každý jmenný prostor je identifikován pomocí Uniform Resource Locator (URI), který vypadá jako (a často je) dobře známý Uniform Rerouce Locator (URL). Jmenný prostor, ve kterém je element definován, může být určen pomocí věci pojmenované atribut. Atributy se objevují uvnitř otevíracího tagu elementu a mohou obsahovat informace o vlastním elementu (jsou používány pro více účelů než jsou jmenné prostory, jak uvidíme dále v kapitole). Například v elementu

```
<qb:Ucet xmlns:qb= http://www.qwickbank.com/bank >
   729-1269-4785
</qb:Ucet>
```

atribut xmlns v otevíracím tagu elementu označuje, že element Ucet je definován ve jmenném prostoru identifikovaném pomocí URI `http://www.qwickbank.com/bank`. Také definuje zkrácené jméno pro tento jmenný prostor, qb, které může být použito pro označení ostatních elementů obsažených v jeho těle (ačkoli v příkladu žádné takové nemáme).

XML samozřejmě obsahuje mnohem více než těchto pár popsaných myšlenek. Základní pochopení webových služeb však nevyžaduje o moc víc znalostí, než jsou základy toho popisného jazyka dat. Vyzbrojeni znalostmi základních koncepcí jsme připraveni k prozkoumání specifických technologií webových služeb.

DEFINUJEME ROZHRANNÍ WEBOVÉ SLUŽBY: WSDL

Pro vyvolání webové služby musí klientský software znát několik věcí. Jméno operace, kterou si přeje vyvolat, společně s informací o typech vstupních parametrů operace a typu výsledku, který je vrácen. Aby se vývojář klientské aplikace a serverového software mohli domluvit o těchto věcech, musí existovat jistý standardní formát, v němž může být informace – rozhranní mezi klientem a serverem - popsána. V COM, například, mohou být rozhranní a operace popsány v Interface Definition Language (IDL). Webové služby potřebují něco podobného.

Pro vyřešení problému byl vytvořen WSDL. Na rozdíl od COM IDL, jehož syntaxe je odvozena od programovacího jazyku C, WSDL je definován pomocí XML. Úplné porozumění definici, třeba i malého rozhranní vyjádřené ve WSDL, vyžaduje docela dobrou znalost XML, byť základní elementy WSDL definice jsou jednoduše pochopitelné.

Následující příklad ukazuje základní komponenty velmi jednoduchého WSDL rozhranní. Toto rozhranní obsahuje pouze jednu operaci, DejZustatek, která požaduje číslo účtu jako vstupní parametr a odpoví hodnotou zůstatku na účtu. (Co následuje není platný WSDL, ani platný XML. Jde pouze o zjednodušenou ilustraci klíčových aspektů WSDL.)

```
<definitions name=AccountAccess >
   <types>
     <element name=BalanceRequest >
       <-- definice vstupních typù, jako je Account -->
     </element>
     <element name=BalanceResult >
     <-- definice výstupních typù, jako je Balance -->
     </element>
   </types>

   <message name=GetBalanceInput >
     <part name=body element=BalanceRequest />
   </message>
   <message name=GetBalanceOutput >
     <part name=body element=BalanceResult />
   </message>

   <portType name= AccountAccessPortType >
     <operation name= GetBalance >
       <input message= GetBalanceInput />
       <output message= GetBalanceOutput />
     </operation>
   </portType>

   <binding name= AccountAccessSoapBinding
       type= AccountAccessPortType >
     <soap:binding
     transport= http://schema.xmlsoap.org/soap/http />
     <operation name= GetBalance >
       <-- definice pro pøipojení vstupù a výstupù -->
     </operation>
   </binding>
```

```
<service name= AccountAccessService >
      <port name= AccountAccessPort
               binding= AccountAccessSoapBinding>
        <soap:address

location= http://www.qwickbank.com/accounts />
      </port>
    </service>
</definitions>
```

Dokonce ani tento zjednodušený popis primitivního rozhranní není tak jednoduchý. S malým úsilím však nebude těžké pochopit, které z důležitých částí definice rozhranní jsou a které nejsou přítomny.

Abychom označili celkovou strukturu rozhranní, stejně jako u každého WSDL rozhranní, všechny elementy v tomto příkladu jsou obsaženy uvnitř kořenového elementu `definitions`. Klíčovými elementy uvnitř `definitions` jsou `types`, `messages`, `portType`, `binding` a `service`. Jak bude každý z nich použit se dozvíme za chvíli. Nejdříve je nutné si uvědomit, že obsah WSDL rozhranní lze rozdělit do dvou logických částí. Nejdříve jsou abstraktně definovány operace pomocí elementů `types`, `message` a `portType`. Jakmile jsou definice provedeny, abstraktní operace jsou připojeny ke specifickým protokolům pomocí elementů `binding` a `service`. WSDL nepředpokládá použití žádného konkrétního protokolu pro vyvolání operací, které popisuje a tak stejná množina operací může být připojena k více než jednomu protokolu.

Podívejme se ještě jednou do příkladu na elementy `types`, `message` a `portType`, abychom zjistili, jak jsou operace specifikovány. Element types je použit k definici základních typů, které budou potřebné v dalších částech definice. V uvedeném příkladu to jsou typy `BalanceRequest`, který představuje číslo účtu a `BalanceResponse`, představující typ návratové hodnoty. Další elementy používají `message` element k definici zpráv zasílaných a přijímaných operací `GetBalance`. První, `GetBalanceInput`, obsahuje hodnotu typu `BalanceRequest`, která již byla definována. Druhá zpráva, `GetBalanceOutput`, obsahuje hodnotu typu `BalanceResponse`.

Jakmile jsou potřebné zprávy a typy, které obsahují, definovány, je možné definovat operaci `GetBalance` založenou na těchto zprávách. Operace jsou specifikovány uvnitř elementu `portType` a jak je uvedeno výše, každá operace může definovat vstupní a výstupní zprávy. V našem případě jsou těmito zprávami `GetBalaceInput` a `GetBalanceOutput`, obě definované dříve.

Všechny tři až doposud popsané elementy se zaměřily na definici operace `GetBalance`. Poslední dva elementy v tomto příkladu propojují (neboli svazují) operaci s konkrétním protokolem, kterým je v našem případě SOAP. Element `binding` definuje `AccountAccessSoapBinding`, explicitní propojení operace `GetBalance` s protokolem SOAP. Na závěr element service explicitně definuje port, spojením `AccountAccessSoapBinding` s konkrétním URL, na kterém může být služba - operace `GetBalance` – nalezena. V našem příkladu je operace GetBalance poskytována fiktivní QwickBank, a proto je přístupná na URL www.qwickbank.com/accounts.

Vše, co jsme viděli, se může zdát jako hodně pracné (a pamatujete, několik detailů bylo vynecháno pro zjednodušení příkladu), ale výsledkem je zcela pružný nástroj pro definici rozhranní webových služeb. S vědomím různorodosti použití webových služeb je tato flexibilita jak potřebná, tak užitečná.

PŘISTUPUJEME K WEBOVÉ SLUŽBĚ: SOAP

Sluší se říci, že základní volbou pro webové služby je SOAP, i když WSDL nepředpokládá žádný konkrétní protokol. SOAP není nijak významně komplikovaný, a tak i velmi krátký popis nevynechá příliš detailů.

Na počátku si uvědomme, co je potřeba pro vyvolání webové služby. Pro přenesení libovolného typu dotazu a navrácení odpovědi bude užitečné definovat obálku, která ponese informace o těchto zprávách. Klíčovou částí SOAP je tedy definice formátu standardní obálky. Každá zasílaná zpráva většinou obsahuje jistý typ informace. Volání operace GetBalace, popsané v předchozích odstavcích, obsahuje řetězec reprezentující číslo účtu. SOAP také specifikuje množinu kódovacích pravidel pro reprezentaci přenášených dat. Protože je běžné vyjádřit vzdálenou komunikaci jako dotaz následovaný odpovědí, SOAP definuje konvenci pro reprezentaci volání vzdálené procedury a její odpovědi. (Zapamatujte si však, že SOAP může být také použit pro asynchronní komunikaci – není omezen pouze na synchronní volání).

Tyto tři věci – obálka, sada kódovacích pravidel a volitelná konvence vzdáleného volání – jsou podstatou toho, co SOAP specifikuje. Je zde několik věcí, které SOAP nespecifikuje a to z důvodů zachování co nejširší použitelnosti a zabránění opakovanému vynalézání existujících technologií. Specifikace nedefinuje, jaký protokol by měl být použit pro přenesení SOAP zpráv, i když ve skutečnosti definuje jednu z možností: přenesení SOAP dotazu a odpovědi přes HTTP. Mohou být však použity i ostatní protokoly. Jak bylo zmíněno již dříve, je například zcela bezproblémové přenést SOAP zprávu přes SMTP nebo jinou messaging technologii.

SOAP specifikace také nedefinuje, jak jsou realizovány služby bezpečnosti jako je autentikace a privátnost dat. Tyto služby jsou místo toho podle potřeby definovány uvnitř protokolů přenášejících SOAP zprávy. Při transportu zpráv po HTTP kanálu bude jednou ze zřejmých variant použití protokolu Secure Socket Layer (SSL), který vytváří bezpečný kanál. SOAP zprávy poslané pomocí jiných protokolů, jako je technologie front zpráv (message queuing), mohou používat jiné mechanismy pro realizaci požadovaných bezpečnostních služeb.

Protože SOAP je běžně přenášen přes HTTP, může projít přes firewall téměř nedetekovaně. Firewall většinou nechává port 80, používaný pro HTTP, otevřený pro přístup k webovým serverům. Tradiční protokoly pro vzdálený přístup jako Distributed COM (DCOM) a Internet InterORB Protocol (IIOP) skupiny Object Management Group, požadují obecně otevření celé řady portů na firewallu, něco, co administrátoři sítě velice neradi vidí. Použitím portu 80 SOAP odstraňuje tento problém. Administrátoři mohou i přesto blokovat SOAP volání, odfiltrováním HTTP dotazů, které obsahují SOAP zprávy.

Stejně jako u WSDL je nejlepším způsobem, jak získat dobrou představu o SOAP, pohled na příklad. Předpokládejme, že si klient přál přístup k operaci GetBalance vystavené bankou QwickBank, jak jsme definovali v předchozí kapitole. SOAP zpráva pro tuto operaci bude vypadat následovně:

```
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV = http://schemas.xmlsoap.org/soap/envelope/
        SOAP-ENV:encodingStyle =
                http://schemas.xmlsoap.org/soap/encoding/ >
        <SOAP-ENV:Body>
                <qb:GetBalance xmlns:qb= http://www.qwickbank.com/bank >
                        <Account>729-1269-4785</Account>
                </qb:GetBalance>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Obdobně jako jiné SOAP zprávy se i tato skládá z elementu Envelope, který obsahuje element Body. Envelope začíná identifikací standardního XML jmenného prostoru, který sám definuje SOAP Envelope a dále styl kódování, použitý uvnitř Body. Neexistuje žádný striktní požadavek, aby se vyskytovaly v každé zprávě, ale oba jsou zde často zahrnuty.

Body obsahuje hlavní část zprávy a také ilustruje SOAP konvenci pro vytvoření vzdáleného dotazu. V tomto případě je operace GetBalance vyvolána a definována v jiném XML jmenném prostoru, vytvořeném QwickBank bankou. Jak již bylo řečeno, operace má jeden parametr, číslo účtu, jehož zůstatek má být vrácen. V našem případě to je 729-1269-4785.

Odpověď na uvedenou zprávu by mohla vypadat následovně:

```
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV = http://schemas.xmlsoap.org/soap/envelope/
        SOAP-ENV:encodingStyle =
http://schemas.xmlsoap.org/soap/encoding/ >
        <SOAP-ENV:Body>
                <qb:GetBalanceResponse
                 xmlns:qb= http://www.qwickbank.com/bank >
                        <Balance>3,822.55</Balance>
                </qb:GetBalanceRensponse>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Ještě jednou. Zpráva se sestává z elementu Envelope, obsahujícího element Body a začíná odkazem na standardní SOAP jmenný prostor. Body obsahuje odpověď vrácenou operací GetBalace, což je objem zůstatku na daném účtu. A stejně jako předtím, XML jmenný prostor, obsahující definici operace, je identifikován uvnitř zprávy v elementu Body.

Uvedené dvě zprávy poskytují velice jednoduchý příklad všech třech klíčových aspektů SOAP: obálku, kódovací pravidla a konvenci pro reprezentaci vzdálených volání a odpovědí. Je zde však mnoho dalších aspektů reálné komunikace, které zde vidět nejsou. Nejzřetelnější z nich je, že z uvedených příkladů není v žádném případě možné určit, jaké zprávy budou přeneseny mezi stranami účastnící se komunikace. Nejběžnější volbou je v současnosti poslání požadavku přes HTTP POST zprávu, s odpovědí přijatou přes standardní HTTP odpověď na typ POST. Existují zde i další možnosti, jak již bylo zmíněno. Ve skutečnosti je možné použít pro vyvolání webových služeb i jiné protokoly než je SOAP. Pro služby s jednoduchým typem parametrů lze použít samotný HTTP, zatímco SMTP lze kombinovat s Multimedia Internet Mail Extension (MIME), ale i jinými volbami.

SOAP specifikace také definuje další užitečné komponenty SOAP zpráv. Obálka může obsahovat například element Header, umožňující zasílání aplikačně specifických informací. Lze jich využít při posílání identifikátoru transakce, autentikačních informací, včetně identity odesílající strany nebo čehokoli, co je pro aplikace užitečné. Specifikace také definuje standardní element Fault pro přenos chybových informací.

Závěrem si ještě jednou zopakujeme, že nic z uvedeného messaging standardu není spojováno s žádným konkrétním operačním systémem, programovacím jazykem nebo objektovým modelem. Tato skutečnost je jedním z nejdůležitějších atributů technologie. SOAP může být, díky svému univerzálnímu návrhu, přijat všemi stranami v současném softwarovém světě.

NALEZENÍ WEBOVÉ SLUŽBY: UDDI

Až doposud dokumentované technologie se zaměřovaly na popis webových služeb a jejich vyvolání po síti. Stále zůstávají jisté problémy. Jak mohou například firmy vyhledat, co webová služba poskytuje svým partnerům? Jak mohou vývojáři najít WSDL rozhranní, které potřebují k implementaci klientů konkrétní služby? Jak může software, který si přeje vyvolat webovou službu zjistit, jestli je tato služba implementována kompatibilním způsobem? A jak lze tyto informace řídit?

Všechny uvedené problémy oslovuje UDDI. Organizace, které chtějí zveřejnit webové služby si mohou založit UDDI obchodní registraci – XML dokument, jehož formát je specifikován pomocí UDDI schématu. Jakmile je vytvořen, obchodní registrace je uložena v replikované databázi známé pod názvem UDDI obchodní registr (UDDI Business registry). Tyto portály, v současnosti provozované firmami Microsoft, Ariba a IBM, spolu navzájem komunikují a podle potřeby udržují vzájemnou konzistenci uložených informací. Klientský software používá pro přístup k informacím v UDDI registru vždy SOAP přes HTTP[1].

UDDI XML schéma definuje obecnou strukturu pro všechny dokumenty obchodní registrace. Každý obsahuje element `businessEntity`, který dále obsahuje jeden nebo více elementů `businessService`. Každý z nich opět obsahuje jeden nebo více elementů `bindingTemplate`, identifikujících konkrétní tModel. Nejjednodušší cesta, jak se seznámit s technologií vede opět přes jednoduchý příklad.

Obdobně, jako u WSDL, je kompletní příklad elementu `businessEntity` i pro jednoduchou webovou službu mimo rozsah této publikace. Následující řádky se snaží poskytnout základní představu o tom, jak by mohly vypadat informace uložené v obchodním registru pro velice jednoduchou službu banky QwickBank. Co následuje není legálním UDDI definicí – jde pouze o zjednodušený příklad.

```
<businessEntity
       businessKey= E7CD0D00-1827-11CF-9956-444553540000 >
       <name>QwickBank</name>
       <description>Internet Bank</description>
       <contacts>
              <contact>
                     <personName>webmaster</personName>
                     <email>webmaster@qwickbank.com</email>
              </contact>
       </contacts>
       <businessServices>
              <businessService
              serviceKey= 15940A43-1956-63DC-0124-444553540000 >
                     <name>AccountAccess</name>
                     <bindingTemplates>
                            <bindingTemplate bindingKey=
                            9E43F20A-1963-22EC-1053-444553540000 >
                            <accessPoint>
                            http://www.qwickbank.com/accounts
                            </accessPoint>
                            <tModelInstanceDetails>
                                   <tModelInstanceInfo  tModelKey=
                            uuid:F7A90326-EB0D-40E0-F55606BD4804>
                                          <description>
                                   QwickBank balance interface
                                          </description>
                                   </tModelInstanceInfo>
                                   <tModelInstanceInfo
                                          tModelKey=
                     uuid:43CAC139-D822-40B5-A004-82DDDC0A12F2 >
                                          <description>
                                          WSDL binding reference
                                          </description>
                                          <instanceDetails>
                            <-- další informace o WDSL rozhranní -->
                                          </instanceDetails>
                                   </tModelInstanceInfo>
                            </tModelInstanceDetails>
                            </bindingTemplate>
                     </bindingTemplates>
              </businessService>
       </businessServices>
</businessEntity>
```

Element `businessEntity`, kontejner pro všechny ostatní, má atribut businessKey, který je nastaven na globálně jedinečný identifikátor přiřazený konkrétní firmě. V první verzi UDDI jsou těmito hodnotami Universal Unigue Identifiers (UUID) - globálně jedinečné 16-ti bajtové hodnoty. `businessEntity` také běžně obsahuje informace o organizaci popsané touto položkou registru. V příkladu obsahuje informace jméno organizace, co dělá a koho kontaktovat pro získání dalších podrobností. Každá část dat používá odpovídající tag: `<name>`, `<description>` a případně `<contacts>`. Každá položka `businessEntity` může obsahovat element `businessServices`, indikující jaké webové služby organizace nabízí.

Element `businessServices` následovně obsahuje jeden nebo více elementů `businessServices`, každý popisující konkrétní službu. Příklad demonstruje pouze jeden `businessService`, který popisuje velmi jednoduchou nabídku banky QwickBank. Obdobně jako `businessEntity` i `businessService` má jedinečný UUID klíč pro identifikaci definice specifické služby. Může také obsahovat jméno služby, která je v našem příkladu „AccountAccess“ a dále pak element `bindingTemplates`.

Každý element `bindingTemplates` může obsahovat jeden nebo více elementů `bindingTemplate`, každý identifikovaný jedinečným UUID. Jeden výskyt `bindingTemplate` v příkladu obsahuje informaci potřebou pro připojení klienta a použití Webové služby banky. Informace obsahuje přístupový bod, vyjádřený jako URL, na němž lze ke službě banky přistoupit a dále zahrnuje informaci o této službě.

Specifika Webových služeb jsou definována pomocí jednoho nebo více *tModelů*[2]. První `tModel` v `bindingTemplate` ukazuje, že je rozhraní poskytnuté bankou QwickBank navrženo pro účely přístupu k zůstatku účtu, zatímco druhý se odkazuje na WSDL soubor s popisem rozhranní. `bindingTemplate` může obecně obsahovat tolik odkazů na tModely, kolik je potřebných pro úplnou specifikaci služby, kterou definuje.

`businessEntity` v našem příkladu obsahuje pouze jeden `businessService`, který dále obsahuje pouze jeden `bindingTemplate`. Skutečný obchod může samozřejmě nabízet mnoho různých služeb s mnoha potenciálními vazbami u každé z nich. Tento jednoduchý příklad vám přesto dává obrázek k vnořené struktuře, kterou UDDI používá pro uložení informací potřebných pro přístup k webové službě.

UDDI je důležitým kouskem skládanky webových služeb. Bez něj by neexistoval způsob, jak by organizace poskytující webové služby dávaly potenciálním uživatelům vědět, co nabízejí a neexistoval by ani jednoduchý způsob, jakým by se klienti dozvěděli potřebné detaily pro přístup k těmto webovým službám. Ačkoli bylo primárním cílem UDDI usnadnit Internetovou B2B integraci, může být také velmi užitečný uvnitř jedné organizace. Webové služby jsou významným pokrokem, který by nemohl bez služeb, jako je UDDI, nastat.

[2] Písmeno „t“ neznamená nic konkrétního. Jméno bylo zvoleno pouze proto, že se UDDI výbor byl schopen na něm shodnout.

WEBOVÉ SLUŽBY NA .NET FRAMEWORK

.NET Framework poskytuje solidní podporu pro implementaci webových služeb. Nejdůležitějším aspektem této podpory je ASP.NET, následovník Active Server Pages. Pomocí ASP.NET mohou vývojáři napsat jednu nebo více .asmx stránek, přičemž každá z nich obsahuje spustitelný kód. Protože .NET Framework podporuje více jazyků, kód lze napsat ve Visual Basic.NET, C# nebo nějakém jiném jazyce. Zveřejnění metody jako webové služby je velice snadné, nezávisle na tom, jaký jazyk si vývojář zvolí. Jediné, co se požaduje, je umístění standardního atributu WebMethod před definici metody. Nemusíme psát WSDL rozhranní nebo vytvářet kód pro přijetí SOAP volání - .NET Framework realizuje tyto služby automaticky za nás. Visual Studio.NET obsahuje také podporu pro přístup k UDDI obchodnímu registru a zjednodušuje vývojářům proces vytváření klientů pro konkrétní webové služby.

K webovým službám postaveným nad ASP.NET není nutné přistupovat přes čistý HTTP-SOAP. V takovém případě však musí být předávané parametry operacím relativně jednoduché a výsledky jsou vráceny v jednoduchém XML dokumentu. Použití SOAP je mnohem pružnější, ale je pravděpodobné, že někteří klienti budou shledávat čisté HTTP volání webové služby mnohem praktičtější, alespoň po určitou krátkou dobu. Ať použijete kterýkoli protokol, vytváření aplikací webových služeb na platformě ASP.NET není vůbec obtížné. Je pravděpodobné, že se jich v blízké době plno objeví.

ZÁVĚR

Webové služby jsou bez debat dobrou myšlenkou. Metoda pro zveřejnění služeb podporovaných mnoha výrobci, běžících na všech typech platforem v prostředí intranetu i Internetu má obrovský potenciál pro otevření světa zcela novým způsobem. Technologie stojící v pozadí Webových služeb - XML, WSDL, SOAP a UDDI - jsou relativně nové. I přes jejich současný zrod jsou podporovány více než jedním výrobcem a vše nasvědčuje tomu, že sehrají významnou úlohu v budoucnosti distribuovaných výpočtů.

BIBLIOGRAFIE

Kapitola 1
Posun typových řešení k distribuovaným výpočtům po Internetu
Daniel Rubiolo, J.D. Meiner, Edward Jezierski, Alex Mackman

Kapitola 2
Co je to Microsoft .NET?
Daniel Rubiolo, J.D. Meiner, Edward Jezierski, Alex Mackman

Kapitola 3
Proč Microsoft .NET?
Daniel Rubiolo, J.D. Meiner, Edward Jezierski, Alex Mackman

Kapitola 4
XML, webové služby a .NET Framework
Andrew Layman a John Montgomery

Andrew Layman je XML šéfarchitekt ve firmě Microsoft. John Montgomery je vedoucím produkt manažerem pro webové služby a .NET Framework ve firmě Microsoft.

Kapitola 5
Úvod do webových služeb: WSDL, SOAP a UDDI
David Chappell

*David Chappell (**david@chappellassoc.com**) je ředitelem Chappell & Associates, výukové a konzultační firmy zaměřené na podnikové softwarové technologie. Tato kapitola byla upravena podle jeho knihy o .NET, publikované Addison-Wesley v roce 2001.*