ClickOnce

technologie pro nasazení aplikací

Borek Bernard, leden 2006

vypracováno jako seminární práce předmětu IZI449 – Distribuované informační a komunikační systémy

Obsah

1 Úvod

Práce se věnuje technologii pro nasazení aplikací ClickOnce. V této kapitole je uveden stručný úvod do problematiky vývoje a nasazení aplikací obecně a zbytek práce se potom věnuje tomu, jak se ClickOnce snaží současné problémy řešit.

1.1 Dilema dnešního vývoje

Aplikace lze dělit podle mnoha různých kriterií, ale v poslední době hraje zvláště významnou roli dělení na aplikace webové a klasické, někdy nazývané desktopové, klientské nebo i jinak (obzvlášť pěkným výrazem je „tlustý klient“). Obě skupiny mají své výhody, ale také určitá omezení. Jaké jsou základní charakteristiky obou skupin?

Začněme webovými aplikacemi, módním hitem dneška. Stále častěji se mluví o „Webu 2.0“, který má zásadním způsobem změnit celou naši práci s počítačem. Co je tedy na webových aplikacích tak přitažlivé? Zaprvé že běží na všech počítačích, kde je přístupný internet. To je tak silná výhoda, že často zastiňuje řadu dalších nevýhod, které jsou s webovými aplikacemi nerozlučně spjaty – jmenujme třeba náročný vývoj, závislost na síti nebo obtížně použitelné uživatelské rozhraní. (Pozn.: často lze slyšet, že jednotnost uživatelského rozhraní patří mezi velké výhody webových aplikací, protože Internet Explorer nebo i další webové prohlížeče vypadají na všech počítačích stejně, ale je nutno si uvědomit, že rozhraní webové aplikace není totéž co rozhraní prohlížeče.) Vedle těchto nevýhod ale existuje i jedna další zásadní výhoda: webové aplikace lze velmi snadno nasadit, případně aktualizovat na novější verzi. Microsoftu trvalo měsíce, než aspoň rozumné procento uživatelů přešlo na Service Pack 2, zatímco když se Google rozhodne, že novou verzi jeho mailu budou všichni uživatelé používat už zítra, nic mu v tom nezabrání. Webový vývoj tak může být mnohem agilnější, což zákazníci oceňují.

Na druhé straně stojí klasické desktopové aplikace. Zatímco u webových aplikací uživatel používá tu verzi, která je aktuálně spuštěna na serveru a vždy je tedy zaručeně aktuální, uživatel desktopové aplikace pracuje s instalací, která byla jednorázově provedena někdy v minulosti a neexistuje žádná záruka, že se jedná o verzi nejnovější. Klasické aplikace mají ještě několik dalších nevýhod, jako např. větší náročnost na lokální zdroje nebo problémy s různými verzemi knihoven (DLL hell), ale to není pro téma práce příliš zajímavé. Důležité je vedle nedostatků vyjmenovat také některé zásadní výhody, jako např. vysoký uživatelský komfort (pro který má angličtina hezký termín „rich user experience“), vysokou produktivitu vývoje nebo rychlou odezvu aplikací. Právě vysoká použitelnost bude i do budoucnosti jedním ze zásadních argumentů, proč nebude žádoucí mít třeba celou kancelář postavenou na webových aplikacích (nechme stranou otázku, jestli to vůbec bude v dohledné době technicky možné).

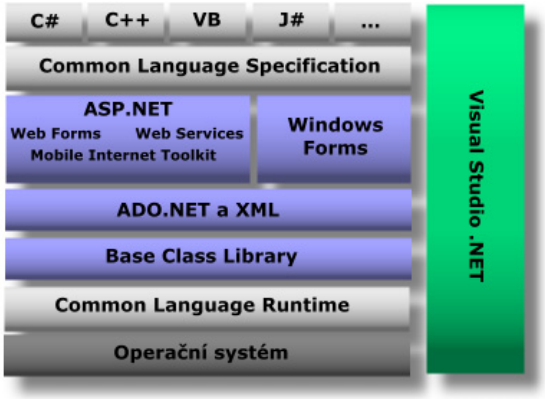
Vypadá to tedy, že je potřeba rozhodnout se mezi webovým prostředím, čímž bude aplikace dostupná na každém počítači v aktuální verzi, ale bude nabízet pouze relativně chudé uživatelské rozhraní, nebo zda vyvinout klientskou aplikaci, která může uživateli nabídnout výrazně komfortnější práci, ale za cenu ztráty kontroly nad aktuálností verze a omezenou online funkčností.

1.2 Co nabízí model chytrých klientů

Odpovědí Microsoftu na právě popsanou neradostnou volbu je model tzv. chytrých klientů, anglicky smart clients. Jedná se o pokus využít výhody obou druhů aplikací a sjednotit je v „nový“ typ, který by nejen sjednocoval výhody, ale také překonával jednotlivé nedostatky.

Na tomto místě je potřeba stručně zmínit .NET Framework, který má pro celý model chytrých klientů i technologie ClickOnce zásadní význam. .NET Framework je relativně novou platformou, do které Microsoft vložil velké investice a která se stává základem jeho současných i budoucích projektů (.NET bude kupříkladu důležitou součástí nových Windows Vista). Jeho základních

komponent je několik. Nejnižší úrovní je Common Language Runtime, běhové prostředí pro tzv. spravovaný (managed) code. Nad ním leží základní knihovny, které poskytují klíčovou funkčnost .NET programům – jedná se především o Base Class Library (základní funkce pro práci s řetězci, obrázky, ale také třeba pro přístup k síťovým prostředkům, k šifrování apod.), potom následuje vrstva pro přístup k datům (ADO.NET a XML), ASP.NET (infrastruktura a třídy pro běh webových aplikací) a konečně Windows Forms, množina tříd pro tvorbu klasických uživatelských rozhraní.



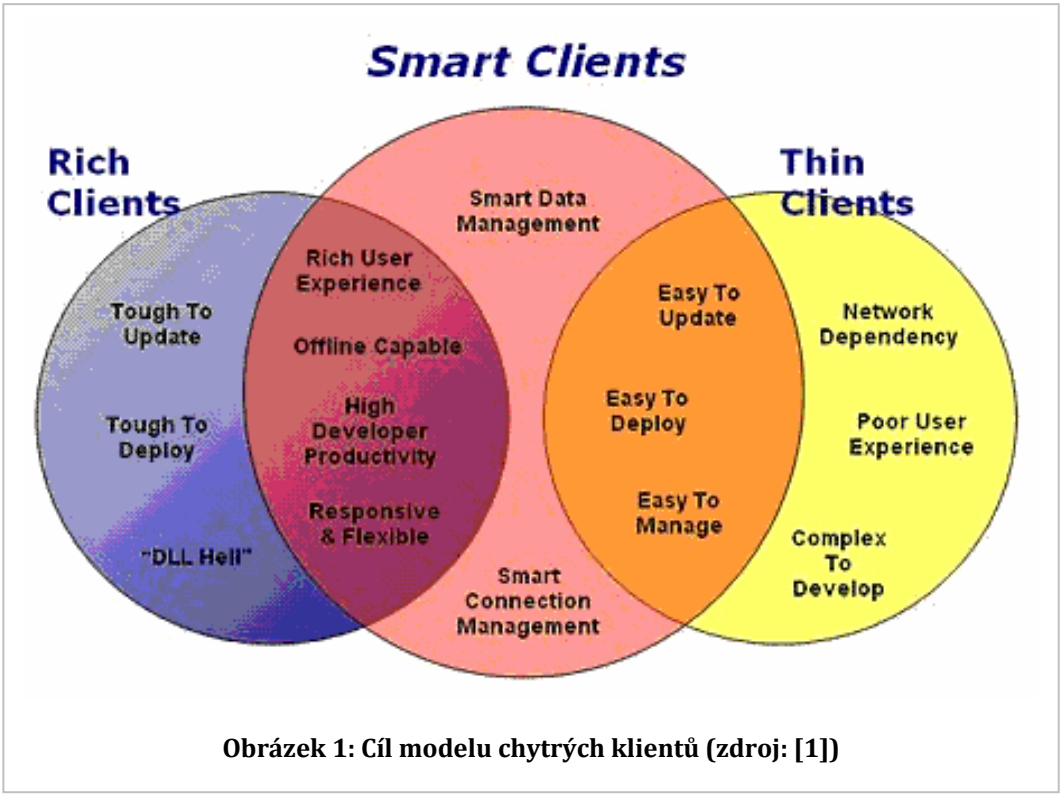
Nad těmito knihovnami leží Common Language Specification, která definuje základní požadavky na jednotlivé jazyky cílené na platformu .NET (jazyková nezávislost je jednou z unikátních vlastností .NET Frameworku), no a nejvýše jsou už potom jednotlivé jazyky jako takové. Několik jich podporuje přímo Microsoft (C#, C++, J#, Visual Basic .NET a JScript), ale desítky jich existují od třetích stran (Pascal, COBOL, Eifell, PHP, Python atd.).

Jak do .NET Frameworku zapadá koncept chytrých klientů? Ve své podstatě je smart client obyčejnou Windows Forms aplikací (Windows Forms je podmnožina .NET Frameworku určená pro tvorbu tlustých klientů, ne nutně Windows aplikací, jak by mohl název evokovat), která je však doplněna systémem nasazení zvaným ClickOnce, který se snaží překonávat typické nedostatky klientských aplikací a o kterém pojednává samostatná část této práce.

Mezi základní charakteristicky chytrých klientů tak patří:

- uživatelský komfort jako v jiných desktopových aplikacích
- napojení na web (stahování a instalace aktualizací, možnost práce online, využívání webových služeb apod.)
- možnost pracovat i offline
- snadné nasazení a aktualizace verzí
- nízký vliv instalace na cílový systém
- usnadnění práce pro vývojáře, správce i uživatele (model webových aplikací ulehčuje především administrátorovi)

Schematicky vyjádřeno vypadá model chytrých klientů následovně:



Obrázek 1: Cíl modelu chytrých klientů (zdroj: [1])

Výhody konceptu chytrých klientů jsou tak poměrně evidentní. O „duši“ tohoto konceptu, technologii nasazení ClickOnce, pojednává druhá kapitola této práce.

1.3 Problémy klasické instalace

Proč nevyhovuje současný model distribuce a instalace aplikací? Jaké jsou jeho nejviditelnější slabiny?

Klasická instalace na platformě Windows je relativně křehká záležitost, která vyvolává řadu otázek, jako třeba jestli je možno instalovat více verzí dané aplikace najednou, jestli instalace aplikace A nepoškodí instalaci aplikace B (tedy jestli mezi nimi neexistuje nějaká závislost na úrovni knihoven) atd. atd.

Windows jsou díky svému návrhu postiženy nepříjemným problémem zvaným DLL hell. Ten je způsoben tím, že sdílené knihovny aplikací jsou instalovány do společného adresáře, resp. adresářů, takže se může stát, že instalace úplně jiné aplikace může knihovnu přepsat její novější verzí, která nemusí být zpětně kompatibilní.

Ale nejen DLL knihovny jsou problémem klasického způsobu nasazení aplikací. Dalším vážným nedostatkem je náročnost instalace na mnoho klientů. Pokud vezmeme v úvahu korporátní prostředí, kde má počítače na starosti IT oddělení, potom instalace aplikace (bez použití nějakého sofistikovaného softwaru na správu nasazení) znamená nutnost fyzicky obejít veškeré počítače a nějaký čas se jim věnovat. A nejen instalace – stejná „obchůzka“ se musí aplikovat i při každé aktualizaci na novou verzi, což může být u některých programů relativně často.

I pokud opustíme korporátní sféru, zůstává problém s aktualizacemi poměrně markantní. Nové verze se vydávají proto, že opravují chyby verzí minulých a přinášejí některé nové funkce, které uživatelé mohou ocenit. U klasických desktopových aplikací je však iniciativa pro vyhledávání aktualizací plně na straně koncového uživatele, což není moc dobře, protože typický uživatel používá počítač na práci a ne k hledání nových verzí svých programů.

Třetím problémem je nižší bezpečnost. Jak už bylo ukázáno na příkladu Service Packu 2 pro Windows XP, jeho přijetí bylo relativně pomalé, ačkoliv přáním Microsoftu by nepochybně bylo, aby k bezpečnostnímu updatu došlo na všech počítačích co nejdříve. Windows jsou možná trochu extrémním případem, protože málokterý produkt je tak rozšířený, ale základní problémy klasické distribuce aplikace tak vynikají ještě zřetelněji.

Po stránce aktuálnosti verzí se jako ideální jeví webový model, který administrátorům umožňuje mít nad používanou verzí plnou kontrolu. To je také jeden z hlavních důvodů, proč slovo intranet končí na „net“ – firmy přišly na to, že investovat větší finanční prostředky do vývoje intranetového portálu (protože je webový vývoj dražší a komplikovanější) se bohatě vrátí při ušetření nikoliv nevýznamných nákladů na údržbu systému.

Faktem nicméně zůstává, že na úrovni dnešních nástrojů a technologií je webový vývoj výrazně komplikovanější než vývoj klasických aplikací. Pro druhou zmíněnou skupinu existují už velmi dlouho poměrně komfortní nástroje (ať už jmenujeme Delphi, Visual Basic, Javu nebo Visual Studio pro .NET), s kódem lze pomocí jiných nástrojů snadno provádět refaktoring, existuje dostatek zkušených vývojářů, mnoho komponent atd. atd. Webový vývoj se také postupně stává snažší. První zásadní kroky v tomto směru podnikl Microsoft se svou technologií ASP.NET, když na webovou platformu přinesl vývoj velmi podobný tomu klasickému, takže nyní lze používat opravdové komponenty, které mají své vlastnosti a události, psaní kódu už není skriptování ve stránce (pomocí slavného stylu spaghetti-code) ale plnohodnotné objektově orientované programování atd. Java přinesla specifikaci JavaServer Faces a několik nástrojů, které se snaží webový vývoj ulehčit podobně, jako to nabízí Visual Studio pro .NET vývojáře. Objevují se nové vysoce efektní i efektivní MVC frameworky, jako např. Ruby on Rails. Jednoduše řečeno vývoj na poli webových technologií chvátá kupředu, nicméně zatím ještě tak snadný jako vývoj klasických aplikací není.

Pokud se tedy na věc díváme z pohledu nákladů producenta aplikace, v určitých scénářích by bylo ideální spojit snadnost a levnost vývoje tlustých klientů se snadností a levností nasazení webových aplikací. A to je přesně to, o co se snaží technologie ClickOnce.

2 ClickOnce

Jak už bylo řečeno, celý model chytrých klientů se od klasických desktopových aplikací liší především jiným způsobem distribuce a nasazení. Zatímco u klasických tlustých klientů je potřeba aplikaci nejdříve distribuovat (ať už na CD nosičích nebo třeba přes internet) a potom doufat, že si uživatel časem nainstaluje novou verzi, ClickOnce umožňuje aplikaci jednoduše umístit na webový server a výrazně tak ulehčit problém nasazení a aktualizace verzí.

Jak bylo právě nastíněno, základní vizí ClickOnce je přinést jednoduchost a spolehlivost nasazení webových aplikací na platformu klientských aplikací. Jak se toho snaží dosáhnout, popisuje zbytek kapitoly.

2.1 Z čeho ClickOnce vychází

První kroky směrem ke konkrétní implementaci byly podniknuty už v .NET Frameworku verze 1.x. Vůbec samotná přítomnost spravovaného (managed) běhového prostředí je pro koncept chytrých klientů důležitá, protože umožňuje velmi dobře kontrolovat práva, které má běžící aplikace přidělena. V .NET Frameworku existuje koncept aplikačních domén a izolace aplikací (Application Isolation), takže třeba vůbec není možné provádět útok typu buffer overflow. Bez zacházení do zbytečných technických detailů je možno konstatovat, že jedna aplikace nemá žádný způsob, jak se při svém běhu dostat k datům aplikace druhé. V .NET Frameworku lze navíc nastavovat řadu dalších oprávnění, např. právo přístupu na disk apod.

Další důležitou věcí, kterou v tomto ohledu .NET Framework verze 1 přinesl, byla propracovaná správa verzí. Jak bylo řečeno před chvílí, Windows odedávna trpí problémem DLL hell, což se snažil .NET Framework řešit. Základní spustitelnou jednotkou je v .NET Frameworku tzv. sestavení neboli assembly, což je zjednodušeně řečeno spustitelný kód spolu s metadaty. Ačkoliv má tedy soubor nadále tradiční tvar název.exe nebo název.dll, ke svému běhu už potřebuje .NET Framework, který zajistí just-in-time kompilaci mezikódu (Microsoft Intermediate Language) do cílového strojového kódu daného procesoru, zavedení do paměti a spuštění. Během tohoto procesu bere .NET Framework v úvahu i metadata, která v assembly nalezne, tudíž má plnou informaci o používané verzi (verze je v metadatech povinně přítomna). .NET Framework podporuje v zásadě dvě možnosti umístění assembly – jednak v podadresáři dané aplikace a jednak v tzv. Global Assembly Cache (GAC). Pokud je assembly umístěna v podadresáři aplikace, nehrozí nebezpečí, že by omylem nahradila knihovnu nebo spustitelný kód jiné aplikaci. Ale ani při umístění do centrálního úložiště (GAC) není možné, aby knihovna jiné aplikace byla omylem přepsána, protože v rámci GAC musí mít assembly tzv. silné jméno (strong name), které je tvořeno kombinací prostého textu, čísla verze, informace o kultuře a hlavně digitálního podpisu, což zajišťuje unikátnost assembly v rámci GAC. Jakákoliv kolize je tedy naprosto vyloučena.

Na co naopak .NET Framework verze 1.x nepamatoval, je bezpečnost instalace z webu. Bylo sice možné do stránky vložit běžný <a> odkaz na určitý spustitelný soubor, ale nebyla podporována lokální instalace. Celkově ale první verze Frameworku vytvořila některé důležité základy, na kterých ClickOnce staví.

Z právě řečeného vyplývá, že technologie ClickOnce je vázána na .NET Framework 2.0, který byl vypuštěn v listopadu 2005. Jelikož bylo součástí tohoto vydání i nové Visual Studio 2005, ClickOnce nezůstalo stranou a je plně v rámci vývojového procesu podporováno. Visual Studio 2005 přináší novou strukturu projektových souborů postavenou na technologii MS Build (jedná se o určitou obdobu oblíbeného systému Ant) a ClickOnce je přímo součástí tohoto nového projektového systému. Záležitosti týkající se instalace aplikace tak nejsou věcí, která by se řešila až po ukončení vývoje, ale je jeho integrální součástí. Ve Visual Studiu je dostupný tzv. Publish Wizard, který pomocí několika podporovaných protokolů (FTP, UNC, FrontPage Server Extensions) přenese kód na server a zajistí doplnění příslušných meta-popisných souborů (o těch bude řeč později).

2.2 Porovnání ClickOnce s ostatními přístupy

Zde je stručné shrnutí postavení ClickOnce mezi dalšími dvěma přístupy:

	Web	**ClickOnce**	**MSI klient**
Široká dostupnost	x		
Nasazení „bez doteku“	x	**x**	
Nízký vliv na systém	x	**x**	
Per-user instalace / běh	x	**x**	
Bohaté UI, interaktivita		**x**	x
Offline možnosti		**x**	x
Integrace s prostředím Windows		**x**	x
Sdílené komponenty na cílovém počítači			x
Neomezené možnosti při instalaci			x

Tabulka 1: Srovnání ClickOnce s jinými přístupy (zdroj: [2])

Z této přehledové tabulky lze rychle vyčíst, jaké jsou cíle a ambice ClickOnce, potažmo modelu chytrých klientů. Nesnaží se zcela nahradit webové aplikace a/nebo klasické aplikace, pouze se snaží využít hlavních předností obou světů a nabídnout koncovému uživateli poměrně atraktivní kombinaci vlastností z dosud oddělených množin.

Zajímavé je přímé srovnání s doposud používanou službou Windows Installer:

	ClickOnce	**Windows Installer**
Po-instalační rollback	Ano	Ne
Aktualizace z webu	Ano	Ne
Nemá vliv na sdílené komponenty nebo další aplikace	Ano	Ne
Bezpečnostní oprávnění zaručena	Zaručena jsou pouze oprávnění důležitá pro aplikaci (bezpečnější)	Ve výchozím stavu jsou všechna práva zaručena (méně bezpečné)
Požadovaná bezpečnostní oprávnění	Internetová nebo intranetová zóna (plná důvěra pro instalaci z CD)	administrátorská práva
Podepsání aplikačního a deployment manifestu	Ano	Ne
Uživatelské rozhraní instalace	Jednoduchý prompt	Průvodce
Instalace assemblies (sestavení) na požádání (on demand)	Ano	Ne
Instalace společných komponent	Ne	Ano
Instalace ovladačů	Ne	Ano
Instalace do GAC	Ne	Ano
Instalace pro více uživatelů	Ne	Ano
Přidání zástupce do nabídky Start	Ano	Ano
Přidání aplikace do nabídky Po spuštění	Ne	Ano
Přidání aplikace do seznamu oblíbených	Ne	Ano
Registrování typů souborů	Ne	Ano
Přístup do registrů v průběhu instalace	Omezený	Ano
Patchování binárních souborů	Ne	Ano
Cílová lokace instalace	ClickOnce application cache	adresář Program Files

Tabulka 2: Srovnání ClickOnce a Windows Installer (zdroj: [3])

Tato tabulka poměrně podrobně ukazuje schopnosti jednotlivých technologií a mělo by z ní být zřejmé, že ClickOnce a Windows Installer nejsou příliš konkurenční technologie, protože jejich použití i schopnosti se značně liší. Pro určitý druh aplikací, které musejí být snadno distribuova-

telné přes internet, se velmi hodí ClickOnce a už méně Windows Installer, ale třeba pro takový Microsoft Office je a nejspíš i zůstane jedinou volbou Windows Installer.

2.3 Kdy zvolit Windows Installer a kdy ClickOnce?

Minulá podkapitola naznačila, že ne vždy je ClickOnce optimálním způsobem nasazení. Zde je seznam otázek, které je potřeba si na začátku položit ([4]):

- Potřebuje aplikace vstup od uživatele během instalace?
- Potřebuje konfigurovat Active Directory nebo COM+?
- Pracuje aplikace takovým způsobem, že vytváří soubory, zapisuje do registru nebo obecně vytváří zdroje, které na počítači zůstanou i po odinstalování aplikace?
- Instaluje aplikace COM komponenty?
- Potřebuje registrovat komponenty pro COM Interop?
- Instaluje nějaké služby?
- Je potřeba aplikaci nainstalovat do určitého umístění?
- Je potřeba instalovat assembly do GAC?
- Obsahuje aplikace komponenty, které jsou instalovány v závislosti na operačním systému nebo hostitelském prostředí?

Pokud je odpověď byť jen na jedinou otázku „ano“, bude vhodnější, nebo dokonce nutné použít Windows Installer.

Zajímavou možnost navíc prezentuje článek [8], který popisuje koncept využití Windows Installeru i ClickOnce nasazení současně. Základním ideou je vytvoření klasického Windows instalátoru, který má na straně cílového počítače plná práva a může tak dělat všechny věci, které bude aplikace potřebovat. Jako jeden ze svých kroků pak „navštíví“ stránku pro ClickOnce instalaci, čímž proběhne nasazení podle ClickOnce scénáře.

Idea to je zajímavá, otázkou je, nakolik je toto řešení použitelné v praxi. ClickOnce má své jasně definované použití a není jasné, zda ho lze „zabalit“ do klasického Windows installeru. Například jednou z charakteristik ClickOnce aplikací je instalace nebo běh s nulovým dopadem na cílový počítač (tj. že po odinstalování / zavření aplikace bude uživatelův systém ve zcela totožném stavu jako před používáním aplikace). Jakmile se ale do celé věci přidá Windows Installer, potom také on bude zodpovědný za vyčištění všech zdrojů při odinstalaci a veškerá zodpovědnost se tak přesouvá na tvůrce instalačního balíčku. Mnoho práce tedy ušetřeno není (při použití ClickOnce je odstranění aplikace plně v kompetenci .NET Frameworku).

2.4 Strategie nasazení ClickOnce aplikací

Existují tři různé strategie pro nasazení ClickOnce aplikace:

- instalace z webu nebo ze sítě
- instalace z CD
- spuštění aplikace z webu nebo ze sítě

Při instalaci z webu nebo ze sdíleného síťového disku uživatel navštíví webovou stránku, kde klikne na odkaz (nebo v případě síťového disku na ikonu aplikace). V důsledku toho se aplikace stáhne na cílový počítač, nainstaluje a nakonec spustí. Do Start menu jsou přidáni odpovídající zástupci a rovněž je zaregistrován odkaz do panelu Přidat/odebrat programy.

Toto je výchozí strategie nasazení, předpokládá však poměrně rychlé internetové připojení koncového uživatele nebo dostupnost aplikace přes lokální síť.

Hlavní výhodou instalace z CD je na druhou stranu možnost distribuce aplikace i uživatelům, kteří momentálně nejsou online. Ačkoliv se tato strategie jmenuje „instalace z CD“, aplikaci lze samozřejmě distribuovat na libovolném nosiči, ať už se jedná o CD, DVD nebo třeba flash disky.

Podobně jako v předchozí strategii, také nyní se po uživatelově příkazu k instalaci aplikace nejdřív nainstaluje a potom spustí, přičemž jsou zástupci přidány jak do menu Start, tak do panelu Přidat/odebrat programy.

Poslední strategií je spuštění aplikace přímo z webu nebo ze sdíleného síťového disku. Zde se aplikace chová prakticky stejně jako webová aplikace – uživatel na nějaké stránce klikne na odkaz, aplikace se stáhne a spustí, ale po jejím ukončení na cílovém počítači ani známky po tom, že by zde byla kdy spuštěna. Nedochází k lokální instalaci, k přidání zástupců do menu Start ani do panelu Přidat/odebrat programy.

Tato strategie není příliš častá a je vhodná pouze u aplikací, které jsou spouštěny zřídka. V souvislosti se spouštěním aplikací z webu je ještě potřeba zmínit jednu vlastnost: pokud se bude jednat o rozsáhlou aplikaci o mnoha megabajtech a s velkým počtem knihoven, přenosové pásmo bude šetřeno v důsledku toho, že natahovány jsou pouze ta sestavení (assembly), která jsou aktuálně potřeba.

2.5 Strategie aktualizací ClickOnce aplikací

Vedle strategie pro instalaci ClickOnce aplikací, což se týká pouze první distribuce a spuštění, existuje také několik strategií pro aktualizaci ClickOnce aplikací.

ClickOnce nabízí pohodlnou podporu pro automatickou aktualizaci aplikací. Aplikace periodicky načítá svůj deployment manifest (o něm bude řeč později), aby zjistila, jestli nejsou k dispozici nějaké aktualizace. Pokud ano, nová verze je stáhnuta a spuštěna (ClickOnce se chová inteligentně a stahuje pouze soubory, které byly změněny).

Stejně jako u strategií nasazení, také zde existují tři základní možnosti:

- kontrola nové verze při spuštění aplikace
- kontrola nové verze po spuštění aplikace ve vláknu běžícím na pozadí
- poskytnutí uživatelského rozhraní pro aktualizace

Všechny tři strategie ke své práci potřebují internetovou konektivitu. Pokud ta není přístupna, aplikaci to nijak vadit nebude, ale k hledání aktualizací samozřejmě docházet nebude.

Společná je rovněž možnost určit, jak často má k aktualizacím docházet. Může to být při každém startu aplikace, ale také třeba jednou týdně nebo měsíčně. Pokud v době, kdy by mělo dojít k aktualizaci, není dostupné připojení k internetu, provede se další pokus při příštím spuštění aplikace.

Výchozí strategií je v tomto případě kontrola nové verze po startu aplikace v separátním threadu běžícím na pozadí. Zvolena byla zřejmě proto, že vyhovuje i na pomalém připojení, kde zjištění nové verze, potažmo stažení nových komponent může nějakou chvíli trvat. K oznámení o nové verzi dojde při příštím startu aplikace.

Strategie kontroly při spuštění se může hodit v případě, kdy je instalace nové verze povinností (třeba proto, že se vývojář rozhodl používat jinou webovou službu a stará verze aplikace by tak z principu nemohla fungovat). Pomocí konfigurační sekce v příslušném XML dokumentu (deployment manifestu) lze určit, jaká minimální verze je pro spuštění požadována.

Poslední strategie, tedy tvorba uživatelského rozhraní pro aktualizaci, není ani tak úplnou a samostatnou strategií, jako spíš určitým pomocníkem pro uživatele. Ten tak např. může mít k dispozici tlačítko pro okamžitou kontrolu nových verzí nebo možnost nastavit interval kontroly nových verzí. Tato strategie se hodí v případě, kdy jednotliví uživatelé budou požadovat jiné aktualizační strategie.

Zajímavou možností na závěr, kterou by také bylo možno nazvat strategií, je úplné blokování aktualizací. Může se stát, že určitá jednorázová aplikace nemá být nikdy aktualizována, ale autor chce přesto využít ostatních výhod distribuce pomocí metody ClickOnce. I toto je možné (pomocí nastavení „The application should check for updates“ na false nebo pomocí odstranění sekce <Subscription> v deployment manifestu).

2.6 Bootstrapping

V praxi je velmi časté, že aplikace využívá některé další komponenty. Jako příklad je možno uvést program, který ke své práci potřebuje Microsoft Data Access Components (MDAC) pro přístup k datům. Jiným příkladem může být osobní databáze disků CD, která jako úložiště dat používá SQL Server Express (dříve známý jako MSDE, Microsoft Desktop Engine).

Obecně může existovat řada komponent nebo programů, na kterých je aplikace externě závislá. Její instalace před instalací těchto předpokladů (prerequisites) vcelku nemá smysl, protože bez nich stejně nemůže fungovat. Proto je potřeba nějakým způsobem zajistit, aby na počítači předem byly. Možná může mít aplikace štěstí a uživatel už potřebné komponenty na svém počítači má, ale v opačném případě je nutno provést jejich předinstalaci. Proces, který umožňuje potřebné komponenty distribuovat společně se samotnou aplikací a nainstalovat je v případě potřeby, je nazýván bootstrapping.

Jak vypadá podpora bootstrapping u ClickOnce aplikací v praxi? Při nasazování aplikace se Visual Studio zeptá, jestli má přidat také určité komponenty, jako třeba .NET Framework. Pokud ano, je vytvořen soubor Setup.exe, který na cílovém systému zjistí přítomnost potřebných komponent a v případě potřeby je nainstaluje. Visual Studio rovněž generuje soubor Publish.htm, který v typických případech (pokud není řečeno jinak) generuje dva odkazy – jeden na soubor bez bootstrapperu, druhý na verzi včetně bootstrapperu. Pokud je však jedinou závislostí .NET Framework a uživatelův prohlížeč hlásí, že ten už na cílovém počítači nainstalován je, zobrazí se pouze jeden odkaz.

Bootstrapping je pro uživatele příjemná věc, protože nemusí po internetu shánět komponenty požadované k běhu, má ale i jednu malou nevýhodu – řada komponent, kupříkladu .NET Framework nebo SQL Server 2005 Express Edition, budou k instalaci požadovat administrátorská práva nezávisle na tom, jaká úroveň zabezpečení je definována aplikací samotnou. Spíš než o nevýhodu se jedná o vlastnost, se kterou nelze nic dělat, ale je při tvoření bootstrapperu je dobré si uvědomit, že uživatel pravděpodobně bude potřebovat práva administrátora.

2.7 Bezpečnost

Pokud je aplikace instalována a/nebo spouštěna z webu, je klíčovou otázkou bezpečnost takovéto aplikace. Jak již bylo stručně zmíněno dříve, .NET Framework umožňuje poměrně přísně vymezit práva aplikace. Jak konkrétně ona omezení vypadají?

ClickOnce aplikace stažená z internetu běží v prostoru vymezeném pouze jí, přičemž přidělená práva závisejí na bezpečností zóně, ze které byla spuštěna. Zde je přehledná tabulka:

Původ aplikace	**Bezpečnostní zóna**
Spuštěno z webu	zóna Internet
Instalováno z webu	zóna Internet
Instalováno ze síťového disku	zóna Intranet
Instalováno z CD	plná důvěra

Tabulka 3: Práva ClickOnce aplikace

Co když ale aplikace potřebuje vyšší práva, např. proto, aby mohla komunikovat s Excelem nebo s jinými programy? Na to práva ze zóny Internet nestačí. V tom případě je uživatel dotázán, jestli chce povolit zvýšení práv aplikace. Pokud zvýšení práv povolí, problém odpadá, ale pokud to zakáže, instalace vůbec neproběhne (bude roll-backována). Protože je podobné dotazování koncových uživatelů pro ně samotné relativně nepříjemná žáležitost (a ne každý ví, jaká práva by měl jaké aplikaci přidělit), lze na koncový počítač nainstalovat certifikát výrobce a říci, že jeho aplikace mohou pracovat se zvýšenými právy. Zodpovědnost se tak přesouvá na správce, který má o bezpečnosti lepší přehled, na druhou stranu zase vyvstává problém, jak certifikát na koncové stanice doručit. Ale s tím se nedá nic dělat, je naprosto klíčové, aby aplikace ze zóny Internet měly poměrně nízká práva.

2.8 Interní soubory ClickOnce

Pokud vývojář pracuje ve Visual Studiu 2005, je otázka publikování velmi snadnou záležitostí. Stačí spustit průvodce, který se zeptá, kam má aplikaci nasadit a na několik dalších možností. Co když ale Visual Studio k dispozici není?

Technologii ClickOnce lze samozřejmě používat i bez Visual Studia. Pro .NET Framework verze 2 Microsoft zapracoval na tom, aby bylo možno většinu věcí konfigurovat „ručně“ pomocí (nejčastěji) XML souborů. Veškeré konfigurační soubory webových nebo klasických aplikací jsou proto XML soubory, lokalizační zdroje jsou uloženy v XML, buildovací systém (MSBuild) je XML, soubory projektů jsou XML atd. atd. Není tedy divu, že také ClickOnce lze konfigurovat jako pomocí jednoduchých textových souborů.

Zásadní roli hrají dva druhy souborů: deployment manifest a application manifest.

Deployment manifest existuje pro jednu aplikaci vždy právě jeden a obsahuje např. informaci o aktuální verzi aplikaci nebo odkazy na application manifesty.

Zde je příklad:

```
<?xml version="1.0" encoding="utf-8"?>
<asmv1:assembly xsi:schemaLocation="urn:schemas-microsoft-com:asm.v1
assembly.adaptive.xsd" manifestVersion="1.0"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" xmlns="urn:schemas-microsoft-
com:asm.v2" xmlns:asmv1="urn:schemas-microsoft-com:asm.v1"
xmlns:asmv2="urn:schemas-microsoft-com:asm.v2"
xmlns:xrml="urn:mpeg:mpeg21:2003:01-REL-R-NS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <assemblyIdentity name="ValidateChildrenWithConstraints.app" version="1.0.0.0"
publicKeyToken="6c3d632f25ac9964" language="neutral"
processorArchitecture="msil" xmlns="urn:schemas-microsoft-com:asm.v1" />
  <description asmv2:publisher="MS" asmv2:product="Validate With Constraints"
xmlns="urn:schemas-microsoft-com:asm.v1" />
  <deployment install="false" minimumRequiredVersion="1.0.0.0"
trustURLParameters="true">
    <deploymentProvider
codebase="http://localhost/ValidateChildrenWithConstraints.application" />
  </deployment>
  <dependency>
    <dependentAssembly dependencyType="install" allowDelayedBinding="true"
codebase="Debug\ValidateChildrenWithConstraints.exe.manifest" size="5912">
      <assemblyIdentity name="ValidateChildrenWithConstraints.exe"
version="1.0.0.0" publicKeyToken="6c3d632f25ac9964" language="neutral"
processorArchitecture="msil" type="win32" />
      <hash>
        <dsig:Transforms>
          <dsig:Transform Algorithm="urn:schemas-microsoft-
com:HashTransforms.Identity" />
        </dsig:Transforms>
        <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
```

```
        <dsig:DigestValue>RokPpIeGiPcO/+UUi5thetccDTc=</dsig:DigestValue>
      </hash>
    </dependentAssembly>
  </dependency>

  <Signature Id="StrongNameSignature"
xmlns="http://www.w3.org/2000/09/xmldsig#">
    ...
  </Signature>
</asmv1:assembly>
```

Základní prvky deployment manifestu jsou následující:

- **<assemblyIdentity>** obsahuje odkaz na application manifest pro ClickOnce aplikaci
- **<description>** určuje zobrazení v panelu Přidat/odebrat programy
- **<deployment>** je první nepovinný atribut, který určuje parametry nasazení
- **<dependancy>** určuje verzi a umístění odpovídajícího application manifestu
- **<Signature>** obsahuje digitální podpis deployment manifestu

Druhým typem souboru je tzv. application manifest, což je rovněž XML soubor, který určuje nasazovanou aplikaci. Zatímco tedy deployment manifest obsahuje informace o nasazení jako takovém, application manifest obsahuje informace o aplikaci samotné.

Application manifest je specifický pro jednu verzi nasazení. Zatímco tedy deployment manifest existuje pro každou aplikaci vždy právě jeden, application manifestů může být libovolně mnoho. Obvykle jsou ukládány v podadresáři značícím číslo verze.

Zde je příklad application manifestu:

```
<?xml version="1.0" encoding="utf-8"?>
<asmv1:assembly xmlns="urn:schemas-microsoft-com:asm.v2" manifestVersion="1.0"
xmlns:asmv1="urn:schemas-microsoft-com:asm.v1" xmlns:asmv2="urn:schemas-
microsoft-com:asm.v2" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:schemas-microsoft-com:asm.v1 assembly.adaptive.xsd">
  <!--Application Manifest Generated by ClickOnce MG. SCHEMA VERSION 2.0.0.13-
PRE.ADAPTIVE-->
  <asmv1:assemblyIdentity name="DatumCorpApp.exe" version="1.0.0.0"
publicKeyToken="0000000000000000" processorArchitecture="msil" />
  <entryPoint>
    <assemblyIdentity name="DatumCorpApp" version="1.0.0.0"
processorArchitecture="msil" language="neutral" />
    <commandLine file="DatumCorpApp.exe" parameters="" />
  </entryPoint>
  <trustInfo>
    <security>
      <applicationRequestMinimum>
        <PermissionSet ID="FullTrust" Unrestricted="true" />
        <defaultAssemblyRequest permissionSetReference="FullTrust" />
      </applicationRequestMinimum>
    </security>
  </trustInfo>
  <!--Application Files-->
  <file name="DatumCorpApp.exe.config" size="228">
    <hash>
      <dsig:Transforms>
        <dsig:Transform Algorithm="urn:schemas-microsoft-
com:HashTransforms.Identity" />
      </dsig:Transforms>
      <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
```

```
      <dsig:DigestValue>e1diZjAqZT5NNRAxTqoWKSK4iPE=</dsig:DigestValue>
    </hash>
  </file>

  <!--Assemblies-->
  <dependency>
    <dependentAssembly codebase="DatumCorpApp.exe" size="32768">
      <assemblyIdentity name=" DatumCorpApp " version="1.0.0.0"
processorArchitecture="msil" language="neutral" />
      <hash>
        <dsig:Transforms>
          <dsig:Transform Algorithm="urn:schemas-microsoft-
com:HashTransforms.Identity" />
        </dsig:Transforms>
        <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
          <dsig:DigestValue>gybeo+fVPiXh8vsbatWFGx6mtgk=</dsig:DigestValue>
      </hash>
    </dependentAssembly>
  </dependency>
  <dependency>
    <dependentAssembly codebase="DatumCorpHelper.dll" size="33280">
      <assemblyIdentity name="DatumCorpHelper" version="4.0.0.0"
publicKeyToken="e8ed396099c4b4e9" processorArchitecture="msil"
language="Neutral" />
      <hash>
        <dsig:Transforms>
          <dsig:Transform Algorithm="urn:schemas-microsoft-
com:HashTransforms.Identity" />
        </dsig:Transforms>
        <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <dsig:DigestValue>w+C0sOgi3IxbwoGK+IAsoa37z6Y=</dsig:DigestValue>
      </hash>
    </dependentAssembly>
  </dependency>

  <!--Microsoft Common Language Runtime Platform Version Required-->
  <dependency>
    <dependentAssembly preRequisite="true">
      <assemblyIdentity name="Microsoft-Windows-CLRCoreComp"
version="2.0.31121.0" />
    </dependentAssembly>
  </dependency>

  <!--Microsoft Windows Operating System Platform Dependency-->
  <dependency>
    <dependentOS supportUrl="http://www.microsoft.com" description="Microsoft
Windows Operating System">
      <osVersionInfo>
        <os majorVersion="4" minorVersion="10" />
      </osVersionInfo>
    </dependentOS>
  </dependency>
</asmv1:assembly>
```

Je v něm vidět několik základních stavebních kamenů application manifestu:

- <assemblyIdentity> identifikuje hlavní assembly aplikace
- <trustInfo> definuje bezpečnostní požadavky na aplikaci
- <entryPoint> definuje vstupní bod aplikace (kód, který se začne vykonávat po spuštění aplikace)
- <dependency> obsahuje závislosti aplikace, o čemž bylo pohovořeno v kapitole *Boot-strapping*.

- <file> odkazuje na všechny soubory aplikace, které nejsou assembly

Visual Studio vytváření všech těchto manifestů od vývojáře abstrahuje, ale určitě je dobré vědět, že nic není nějakým způsobem maskováno a že s manifesty mohou pracovat i externí nástroje.

3 Shrnutí

Práce měla několik hlavních bodů:

- vysokoúrovňový pohled na to, jaké jsou možnosti vývoje a nasazení aplikací
- popsání základních výhod a nevýhod jednotlivých přístupů
- koncept chytrých klientů jako snaha Microsoftu spojit uživatelský komfort klasických aplikací a snadnost správy a údržby webových aplikací
- představení technologie pro nasazení ClickOnce jako duše modelu chytrých klientů
- konkrétní detaily implementace ClickOnce

Pro určité typy aplikací je tak do budoucna ClickOnce určitě zajímavým způsobem, jak se vypořádat s distribucí a aktualizací desktopových aplikací. Na druhou stranu není model chytrých klientů všelékem – neobsahuje všechny výhody webových aplikací ani všechny výhody tlustých klientů. Do budoucnosti tak budou koexistovat tři typy aplikací: tenký klient, tlustý klient a chytrý klient.

4 Zdroje

[1] *Building Smart Client using .NET*. Dostupné z WWW: http://www.codeproject.com/dotnet/DotNetBuildSmClnts.asp (13. 1. 2006)

[2] Cool, J. *Introducing ClickOnce: The New Application Deployment Model for Windows Forms and "Avalon"*. Dostupné z WWW: http://download.microsoft.com/download/6/6/9/669C56E3-12AF-48C5-AB2A-E7705F1BE37F/CLI370.ppt

[3] Microsoft Corp. *Introduction to ClickOnce Deployment*. Dostupné z WWW: http://msdn.microsoft.com/vbasic/learning/clickonce/ (13. 1. 2006)

[4] Bromberg, P. *ClickOnce Deployment: An Overview*. Dostupné z WWW: http://www.eggheadcafe.com/articles/20051103.asp (13. 1. 2006)

[5] MSDN Library. *Choosing a ClickOnce Deployment Strategy*. Dostupné z WWW: http://msdn2.microsoft.com/en-us/library/71baz9ah.aspx (13. 1. 2006)

[6] MSDN Library. *Choosing a ClickOnce Update Strategy*. Dostupné z WWW: http://msdn2.microsoft.com/en-us/library/s22azw1e.aspx

[7] InstallSite: *Microsoft ClickOnce Technology*. Dostupné z WWW: http://www.installsite.org/pages/en/clickonce.htm

[8] Sanford, M.: Choosing Between ClickOnce and Windows Installer. Dostupné z WWW: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/clickoncevsinstaller.asp

[9] MSDN Library: *ClickOnce Deployment Manifest*. Dostupné z WWW: http://msdn2.microsoft.com/k26e96zf.aspx

[10] MSDN Library: *ClickOnce Application Manifest*. Dostupné z WWW: http://msdn2.microsoft.com/ws1c2fch.aspx