ELSŐ FEJEZET

A .NET 3.5-keretrendszer

Alkalmazásfejlesztésnél a cél mindig ugyanaz: a lehető legjobb szoftver létrehozása a legkisebb idő alatt. A felhasználók igényszintje azonban állandóan, emelkedik. A platformfejlesztők folyamatosan fejlesztenek, hogy kielégítsék ezeket az igényeket, az általuk használt eszközöknek egyre jobbnak és jobbnak kell lenniük.

A .NET-keretrendszer tökéletes példája ennek. A keretrendszer 2002-ben mutatkozott be, 2.0 verziója három évvel később. A 2006-ban bemutatott .NET 3.0-keretrendszer nagy lépést jelentett, számos új technológiával bővült, míg a legújabb verzió, a .NET 3.5-keretrendszer további hasznos kiegészítéseket tartalmaz. Ezzel párhuzamosan előrelépett a Microsoftnak a .NET-alkalmazások készítéséhez használatos legfontosabb eszköze, a Visual Studio 2008 is: legújabb kiadása számos újdonságot kínál a .NET-alkalmazásokhoz. Ezen az úton minden lépés azt célozza, hogy jobb és termelékenyebb környezetet nyújtson annak, aki Windows-szoftvert hoz létre.

Ez az ismertető nagy vonalakban tekinti át a .NET 3.5-keretrendszert. Feltételezzük a .NET-keretrendszer alapszintű ismeretét,így ez a leírás a .NET 3.0- és 3.5-keretrendszerben található új technológiákra fókuszál. Azt szeretnénk bemutatni, hogy ezek, a Windows-alkalmazásokhoz széles körben használt alaptechnológiák napjainkban milyen újdonságokat nyújtanak.

A korszerű alkalmazások felépítésének kihívásai

Egy korszerű alkalmazás létrehozása nem egyszerű feladat: a velük szemben támasztott követelmények alapvető fontosságúak. Az olyan hagyományos követelmények, mint a hatékony üzleti logika létrehozása és a kapcsolat webböngészőn keresztül, még mindig fontosak, de már nem elegendőek. A korszerű alkalmazások új kihívások sokaságát támasztják:

- A felhasználók egyre inkább elvárják, hogy a webes felületek is úgy működjenek, mint a Windows-alkalmazások. Már nem kielégítő, hogy amikor valami változik, új lap töltődik be: túl lassú. *Válaszképes (responsive) böngészőalkalmazások* támogatása vált szükségessé.
- Az adatok a legtöbb alkalmazás esetén középpontban maradnak. Megjelenítési lehetőségeik számottevően bővültek. A relációs adatok még fontosabbak, mivel ők hozzák létre az összerendeléseket az objektumok és kapcsolataik között. Ugyanakkor folyamatosan nő az XML-ben ábrázolt adatok mennyisége, és nem valószínű, hogy ez a trend megváltozik. És, bár ez nem látszik mindig , a futó program objektumai is tartalmaznak adatokat. A következetes adathozzáférést biztosító technológiák révén a fejlesztők kevesebb idő alatt és kevesebb hibával hozzák létre az alkalmazásokat.
- Az alkalmazások általában kommunikálnak más alkalmazásokkal, mind a szervezeten belül, mind azon kívül. A korszerű alkalmazásokat gyakran szolgáltatásorientált architektúrába (Service Oriented Architecture – SOA) is beleillesztik, minek révén bizonyos funkcióit együttműködő szolgáltatásként más alkalmazások számára is elérhetővé teszik. E cél megvalósításához a *szolgáltatásorientált alkalmazások támogatására* van szükség.
- A szervezetek egyre inkább folyamatorientáltan tekintenek saját tevékenységükre. Mivel a legtöbb alkalmazás az üzleti folyamat néhány részét automatizálja, hasznos lehet a folyamat bizonyos lépéseit a kódban explicit módon megadni. Ennek hatásos módja a munkafolyamat-technológia, amely a *munkafolyamat-alapú alkalmazásokat támogató* szemlélet.
- A korszerű felhasználói felület követelményei jelentősen megnőttek. Valós üzleti értékek szolgáltatása gyakran szükségessé teszi, hogy különböző fajtájú dokumentumokkal dolgozzunk, két- vagy háromdimenziós grafikákat használjunk, videókat jelenítsünk meg, és így tovább. Ezek a szükségletek a *különböző felhasználói felületek egységes szemléletével elégíthetők ki*.

Az alkalmazások felhasználóinak gyakran szükségük van arra, hogy valamilyen módon információt szolgáltassanak magukról. A digitális identitás megadására és használatára sok különböző technológia áll rendelkezésre, és gyakran ütköznek olyan problémákba, mint például az adathalászat. A korszerű alkalmazásoknak és felhasználóiknak ezen túl is előnyös lehet a *digitális identitás következetes felhasználói ellenőrzése*.

Tudva azt, hogy napjaink alkalmazásainak gyakran kell megküzdenie ilyen kihívásokkal, az alkalmazások által előállított platformoknak ezek mindegyikére megoldást kellene nyújtaniuk. A .NET 3.5-keretrendszer célja ennek biztosítása Windows-alkalmazások esetében.

A kihívások megoldása: a .NET 3.5-keretrendszer a reflektorfényben

A .NET 3.5-keretrendszer számos olyan technológiát tartalmaz, amelyek a fejlesztők segítségére lehetnek a leírt problémák megoldásában. Néhányuk már része volt a .NET 3.0-keretrendszernek is, míg mások újak a 3.5-ös kiadásban. Ez a rész bemutatja ezek közül a technológiák közül a legfontosabbakat.

ASP .NET AJAX: válaszképes böngészőalkalmazások támogatása

A webböngészők a legnépszerűbb felhasználói felületek az új alkalmazásokhoz. Van azonban egy hagyományos hátrányuk: minden egyes új felhasználói kérés újabb oda-vissza utat igényel a webszerverhez, gyakran új oldal betöltését eredményezi. Sokkal intelligensebb – és gyorsabb – módszer az, amikor az adatokat, amikor csak lehetséges, a háttérben kérjük el, és az oldalnak csak azt a részét frissítjük, amelyik változott. A felhasználók sokkal válaszképesebb alkalmazást láthatnak, mivel kevesebb időt kell azzal tölteniük, hogy várnak az új oldal betöltésére.

Ez tulajdonképpen az, amit az AJAX-szemléletmód jelent a webes alkalmazások felépítésében. A böngésző aszinkron módon előre kéri az adatokat, ahelyett, hogy minden egyes felhasználói kéréshez új oldalt töltene be. A kódot, amely ezt a kérést intézi, tipikusan JavaScriptben írják, és az adat gyakran (de nem mindig) XML-ben van. Ez a három jellemző – aszinkron, JavaScript és XML (Asynchronous JavaScript and XML) – az AJAX név eredete.

Bár az AJAX alapjául szolgáló technológia magja már 1999-ben megjelent az Internet Explorer 5 kiadásában, néhány év kellett, mire közkedveltté vált. Napjainkban az AJAX az új webes alkalmazások domináns technológiája. Ennek megfelelően a .NET 3.5-keretrendszer az ASP.NET AJAX nevű technológiát testesíti meg. Az AJAX-alkalmazások létrehozását a fejlesztők számára az eredeti ASP.NET bővítése teszi könnyebbé.

Nyelvbe ágyazott lekérdezések (Language-Integrated Query): következetes hozzáférés különböző adatokhoz

A legtöbb alkalmazás hasonló módon dolgozik az adatokkal. Az adatok, így a relációs adatbázisok táblái, az XML-dokumentumok és a memóriában tartott objektumok különböző módon jeleníthetők meg. Az adatok egyes típusaihoz való hozzáférés hagyományosan eltérő szemléletet igényel. Az SQL használata a relációs adatokkal való munka esetén gyakori, például, az XML-adathoz az XQuery, a memóriában található adathoz egyedi kód fér hozzá. A cél ugyanaz – információhoz való hozzáférés –, miért nincs hát egységes, általános módszer az összes esethez?

Ezt a célt szolgálja a Language-Integrated Query (LINQ – nyelvbe ágyazott lekérdezés), amely új technológia a .NET 3.5-keretrendszerben. A LINQ nem igényel minden egyes adattípushoz speciális nyelvet és független eljárásokat, ehelyett inkább bővíti a C#-ot és a VB-t, általános elérést téve lehetővé a különböző adatokhoz. Ez a technológia nagy területet fed le, beleértve az objektum/relációs összerendelést és másokat. A cél az, hogy egyszerűbbé tegyük az életet azok számára, akik adattal dolgozó .NET-alkalmazásokat hoznak létre vagy tartanak fenn.

Windows Communication Foundation: szolgáltatásorientált alkalmazások támogatása

A .NET-keretrendszer a kezdetektől módszerek sokaságát nyújtotta a kommunikációhoz. Ezek választéka a következő:

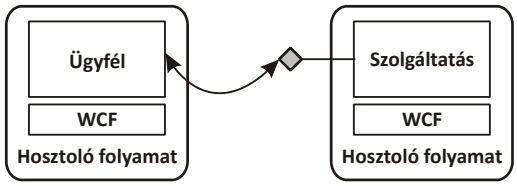
- Az ASP.NET Web Services (ASMX néven ismeretes) együttműködő SOAP-alapú kommunikációt szolgáltat.
- A .NET Remoting a .NET-alkalmazások közötti kommunikációra fókuszál.

- Az Enterprise Services (vállalati szolgáltatások) a méretezhető, tranzakciót használó alkalmazások számára nyújt támogatást.
- A System.Messaging sorba állított üzenetküldéseket támogat a Microsoft Message Queuing (MSMQ) rendszeren keresztül.
- A Web Services Enhancements (WSE), az ASP.NET Web Services kibővítése, több jelenlegi specifikációhoz, például a WS-Securityhoz nyújt támogatást.

Mindegyik technológiának megvan a maga szerepe. Mégis, miért kell több különböző megoldásnak léteznie azonos problémákra? Miért nem hozunk létre helyette egy egyszerű alapot az alkalmazások kommunikációjához?

Tulajdonképpen ez az, amit a Windows Communication Foundation (WCF) tesz. Nem kéri a fejlesztőket arra, hogy minden egyes kommunikációtípus kapcsán más és más technológiát használjanak a különböző alkalmazásinterfészek programozásához, hanem a WCF és egy általános API használatával közös módszert kínál. A Microsoft által jelenleg ajánlott kommunikációs módszer az eredetileg a .NET 3.0-keretrendszer részeként megjelent WCF. A felsorolt technológiák valamelyikét használó alkalmazások többségének a WCF-fel kellene kommunikálnia.

Ahogy az 1.1. ábra mutatja, a WCF alapmodellje egyszerű: az ügyfél hozzáfér néhány szolgáltatáshoz, szükség esetén meghívja függvényeiket. A WCF nem bíz meg semmilyen egyéni hosztot, és így a fejlesztők szabadon használhatják ezt a kommunikációs technológiát bármely hosztfolyamaton belül.



1.1. ábra: *A Windows Communication Foundation alapmodellje*

A WCF a SOAP-on keresztül erős támogatást nyújt az együttműködő kommunikációhoz, fontos része a korszerű számítástechnikának. Támogat néhány WS-* specifikációt, például a WS-Securityt, a WS-ReliableMessagingot és a WS-AtomicTransactiont.

A WCF nem teszi kötelezővé a SOAP használatát, és így más módszereket is használhatunk, például optimalizált bináris protokollt, sorba állított üzenetkezelést MSMQ használatával és az egyszerű, közvetlenül HTTP-re épülő RESTful-eljárást.

Az alkalmazások közötti kommunikáció, legyen bár a szervezeten belül vagy szervezetek közötti, alapvető része a korszerű szoftvernek. A .NET 3.5-keretrendszer ezt a kihívást a WCF szolgáltatásorientált szemléletén keresztül küzdi le.

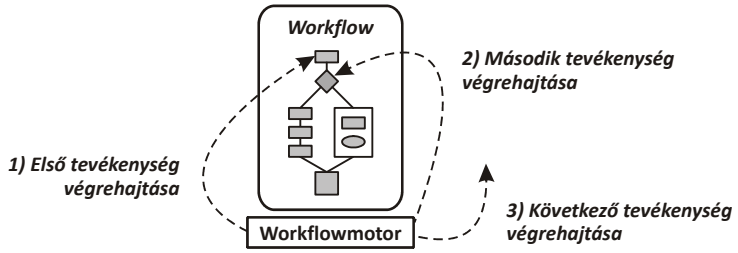
Windows Workflow Foundation: munkafolyamat-alapú alkalmazások támogatása

A munkafolyamat egyszerű ötlet: nem más, mint lépések sorozatának végrehajtása adott sorrendben. Bizonyítható, hogy minden egyes alkalmazás munkafolyamattal dolgozik, mivel minden alkalmazás folyamatokat dolgoz fel. A C# és Visual Basic vagy néhány más programnyelv használatával történő alkalmazásfejlesztés hagyományos szemlélete szerint a lépéseket a folyamaton belül a kódban, implicit módon adjuk meg. Ez természetesen működik, de a folyamatot mélyen beágyazza a program forráskódjába, ezzel a folyamat nehezebben létrehozhatóvá és változtathatóvá válik.

A folyamat-forráskód munkafolyamat-technológiával való létrehozása hatásos módja lehet e probléma kezelésének. A forráskód logikáját nem ömlesztjük szokványos kódba, hanem explicit módon definiáljuk a folyamat minden egyes lépését, majd munkafolyamat-motor segítségével hajtjuk végre. Az eredmény a folyamat tiszta megvalósítása. A Microsoft a Windows Workflow Foundation (WF) segítségével általános munkafolyamat-technológiát kínál a Windowshoz, közös alapot biztosítva a munkafolyamat-alapú alkalmazások építéséhez. Mióta a WF megjelent a .NET 3.0-keretrendszerben, ezt használják az olyan Microsoft-alkalmazásokhoz, mint a Windows SharePoint szolgáltatások, de más szervezetek által létrehozott alkalmazásokhoz is.

Képes-e azonban egyetlen technológia megfelelni azoknak a különböző követelményeknek, amelyeket a különböző munkafolyamat-alkalmazások igényelnek? A WF által elfogadott válasz a munkafolyamat teljesen általános megközelítése. Az 1.2. ábrán látható, hogy a WF-munkafolyamat a WF-motor segítségével adott sorrendben végrehajtható tevékenységek csoportja.

Minden egyes tevékenység tulajdonképpen külön osztály, és bármilyen munkát tartalmazhat, amelyet a munkafolyamatot létrehozó szükségesnek tart. A tevékenységek újra felhasználhatóak különböző munkafolyamatokban, ezzel megkönnyítik automatikus megoldások létrehozását új probléma esetében.



1.2. ábra: *A WorkFlow tevékenységcsoportjai*

A WF általános munkafolyamat-technológiát nyújtva a Windows számára ezt a hasznos szoftverfejlesztési megoldást általánosan elérhetővé teszi a fejlesztők számára. Ahogy a folyamatorientált szoftverfejlesztési szemlélet egyre népszerűbb lesz, a munkafolyamatok használata is valószínűleg növekedni fog.

Windows Presentation Foundation: egységes szemlélet különböző felhasználói felületekhez

A felhasználói felület nagyon fontos része majdnem minden alkalmazásnak. A felhasználók elvárásai jelentősen megváltoztak ezekkel a felületekkel szemben. A hagyományos menüvezérelt grafikus felületek természetesen megmaradnak, de az alkalmazásoknak már szükségük van videók megjelenítésére, animációk futtatására, két- vagy háromdimenziós grafikák használatára és arra, hogy dokumentumok különböző típusaival dolgozzanak. Mindezeknek mindenképpen megvalósíthatóknak kell lenniük akkor is, ha az alkalmazást egyedi gépen vagy webböngészőn keresztül használják.

A Windows szokás szerint különböző módokat nyújt a felhasználói felületek megvalósításához. Például, egy fejlesztő Windows grafikus felületek létrehozásához használhatja a a .NET-keretrendszer részét képező Windows Formsot. Webböngésző-felület létrehozásához HTML- és talán Java-kisalkalmazások vagy JavaScript-kód szükségesek. A videók megjelenítése Windows Media Player vagy valami más alkalmazás használatára támaszkodhat, míg dokumentumformátumok Microsoft Wordben, PDF-ben vagy más módon definiálhatók.

A kihívás a fejlesztők számára világos: nem egyszerű összefüggő felhasználói felületet létrehozni különböző ügyfeleknek, eltérő technológiák használatával.

A Windows Presentation Foundation (WPF), – eredetileg a .NET 3.0 keretrendszerben jelent meg, elsődlegesen ezeknek a kihívásoknak a leküzdésére hivatott. A WPF, azzal, hogy a felhasználói felületek mindezen szempontjai számára következetes technológiai alapot biztosít, egyszerűbbé teszi a fejlesztők életét. A WPF korszerűbb szemléletével, amelybe beletartozik a videók, animációk, két- vagy háromdimenziós grafikák és különböző típusú dokumentumok támogatása, lehetővé teszi, hogy a felhasználók új módon dolgozhassanak az információkkal. Azzal, hogy a WPF közös alapot nyújt az asztali és a böngészős ügyfelek számára, mindkettőjük számára megkönnyíti az alkalmazásfejlesztést.

Az 1.3. ábrán látható képeket, élő grafikákat, háromdimenziós nézeteket és egyebeket tartalmazó példafelület illusztrál néhány dolgot abból, amit a WPF nyújt. Az olyan felhasználói felületeket, mint ez, most konzisztens módon hozhatjuk létre, anélkül, hogy különböző technológiákban jártas fejlesztőkre lenne szükségünk.

A felhasználói felületek létrehozói számára hosszú időn keresztül komoly kihívást jelentett, hogy a hatásos felület létrehozása különböző szereplőket igényelt. A felület mögötti forráskód létrehozásához programfejlesztőkre van szükség, de ők csak ritkán alkalmasak külalak és hangulat meghatározásához. Erre a szerepre tipikusan az ember-gép kölcsönhatáshoz értő szakember a legjobb választás. A régebbi technológiák, mint a Windows Forms, még teljesen a fejlesztőre koncentráltak. Ezek nem igazán hatásosak a fejlesztők és a tervezők együttműködése tekintetében. A WPF az eXtensible Application Markup Language-ra (XAML nyelv) támaszkodik e probléma leküzdésében. A XAML egy XML-alapú nyelv, amely lehetővé teszi a felhasználó felületek deklaratív megadását, kód helyett. Így, a tervező által létrehozott vizuális ábra alapján sokkal egyszerűbbé válik a felületspecifikációk generálása az eszközökhöz, és a velük való munka. Tulajdonképpen, a Microsoft éppen erre a célra kínálja az Expression Blendet. A tervezők ezt az eszközt (és még másokat, amelyeket más gyártók kínálnak) egy interfész kialakítására használhatják, majd megkapják az általuk létrehozott felület XAML-definícióját. A fejlesztők ezt a definíciót beolvassák Visual Studióba, majd létrehozzák a felület által megkívánt forráskódot.

1.3. ábra: *WPF-fel létrehozható felület példája*

Amikor a fejlesztő létrehoz egy közvetlenül Windows alatt futó egyedi WPF-alkalmazást, minden rendelkezésére áll, amit a WPF nyújtani tud. Webböngészőben futó ügyfél létrehozásakor a fejlesztő egy XAML-böngészőalkalmazást hoz létre, általában XBAP-ként hivatkozva rá. Az XBAP, egy letölthető böngésző alkalmazáson belül lehetővé teszi azonos stílusú felhasználói felületek kialakítását ugyanolyan alapokra építve, mint az egyedi WPF-alkalmazásoknál. Az alkalmazás mindkét fajtájához ugyanaz a kód használható, ami azt jelenti, hogy a fejlesztőnek a továbbiakban nincs szüksége különböző ismeretekre az asztali és a böngészőügyfelek felületének kialakításához. Az ilyen gazdag internetalkalmazások jellemzője, hogy egy internetről letöltött XBAP védett tesztágyon (sandbox) fut, ami korlátozza az alkalmazás lehetőségeit. A felhasználói felületek funkcióinak nagy része hozzáférhető XBAP-ban is működő egyedülálló WPF-alkalmazással.

Mind az egyedülálló WPF-alkalmazás, mind az XBAP kihasználhatja a WPF modern grafikai támogatását, beleértve a hardvergyorsítás lehetőségét, a vektorgrafika támogatását stb. A WPF 3D a grafikák készítésének leegyszerűsítésével adatmegjelenítési lehetőségek sokaságát teszi elérhetővé, ami Windows Forms vagy korábbi technológiák alkalmazásával

nem volt lehetséges. A WPF képezi az XML Paper Specification (XPS) alapját is, amely rögzített formátumú dokumentumok megtekintéséhez, megosztásához és nyomtatásához határoz meg szabványos formátumot.

A felhasználói felületek a korszerű alkalmazások komplex és fontos részét képezik. A WPF révén a .NET 3.5-keretrendszer teljesebb és következetesebb megoldást nyújt a felületek jelentette kihívásokhoz. A cél az, hogy hatékonyabbá tegyük a felhasználói felületek létrehozóinak a munkáját, a fejlesztőknek és a tervezőknek egyaránt.

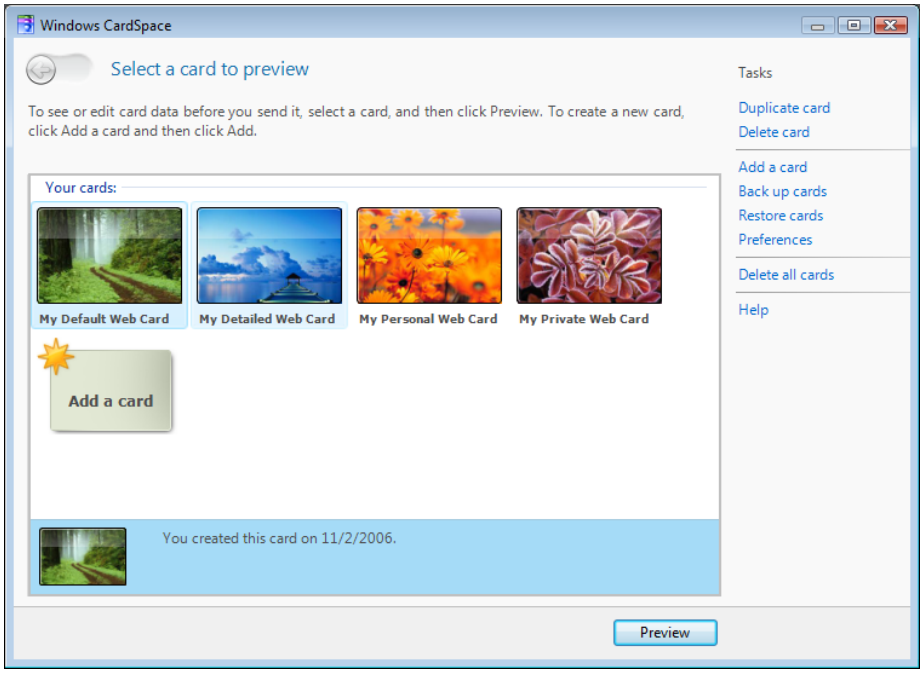
Windows CardSpace: a digitális identitás következetes felhasználói ellenőrzése

Gondoljunk arra, hogyan mutatkoznak be manapság az emberek az interneten. Az esetek nagy többségében a személyek digitális identitását egy egyszerű felhasználó név fejezi ki. Ez az azonosító jelszóval kombinálva e-mail fiókok, webboltok, bankok és más pénzügyi intézmények online elérésére szolgál. Népszerűségük ellenére a felhasználóneveknek és jelszavaknak számos hátrányuk van. A két legfontosabb:

- Az embereknek nehézséget okoz a különböző oldalakon választott különböző felhasználónevek és jelszavak megjegyzése. Sok ember a könnyebb megjegyezhetőség érdekében ugyanazt az azonosítót használja a különböző oldalakhoz, fokozva ezzel a biztonsági kockázatot.
- A felhasználónév, a jelszó és a többi személyes információ adathalászattal ellopható. Az adathalászok megtévesztő e-mailek küldésével elcsábítják áldozataikat, hogy olyan weboldalra jelentkezzenek be, amelyek ugyanúgy néznek ki, például, mint az áldozat bankjának oldala. Az oldalt tulajdonképpen az adathalász felügyeli, így ha az áldozat egyszer beírja felhasználónevét és jelszavát, az adathalász ezt az információt a valódi oldalon álfelhasználóként használhatja.

A probléma komolysága a digitális identitás kezelésének új szemléletét igényli. A Windows CardSpace, amely eredetileg a .NET 3.0-keretrendszerben jelent meg, nagyon fontos része ennek a szemléletnek. A CardSpace minden egyes azonosítót külön információs kártyaként jelenít meg, hogy segítsen az embereknek nyomon követni digitális azonosítóikat. Ha a webhely fogad CardSpace bejelentkezést, a felhasználó, aki megpróbál az oldalra bejelentkezni, egy olyan CardSpace-választót fog látni, mint

ami a lenti ábrán látható. A felhasználó azzal, hogy kiválaszt egy kártyát, egyben egy digitális azonosítót is választ, amelyet az oldalhoz való csatlakozáshoz használ. A különböző kártyák különböző információkat tárolhatnak, lehetővé téve ezzel a felhasználó számára, hogy ténylegesen ellenőrzése alatt tartsa, amit az egyes oldalak tudnak róla.



1.4. ábra: *CardSpace-választó*

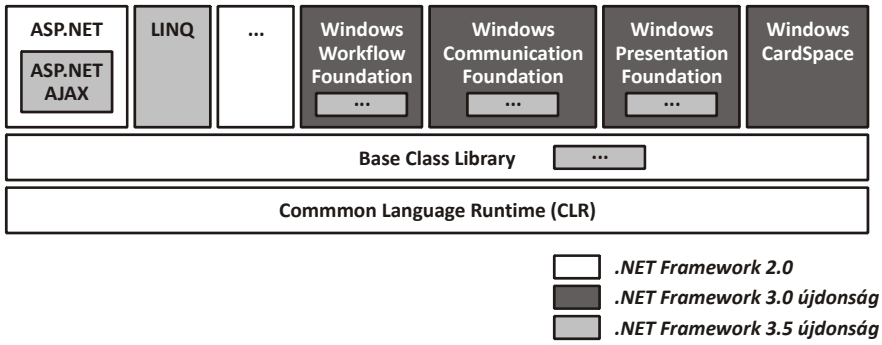
A kártyák által megjelenített azonosítókat egy vagy több azonosítószolgáltató hozza létre. Minden szervezetnek lehet azonosítószolgáltatója, és rendelkezhet magával a CardSpace-alkalmazással is, beleértve az önazonosító szolgáltatót is, amely az ügyfélgépen fut. Ezzel a szolgáltatóval a felhasználók maguk is létre tudják hozni saját azonosítóikat, amelyek nem igényelnek jelszót a hitelesítéshez. A webhelyek, felismerve ezeket az magánkibocsátású CardSpace-azonosítókat, ezekre támaszkodnak a szokásos jelszóalapú szemlélet helyett, csökkentve ezzel a jelszók által létrehozott problémákat.

A Windows CardSpace tulajdonképpen egy nagyobb azonosító metarendszer része. Ez a metarendszer egy teljesen nyitott, publikus protokollra támaszkodva meghatározza a különböző digitális azonosítótechnológiák következetes használatának módját különböző platformokon (beleértve a

Windowson kívül más operációs rendszereket is) és különböző alkalmazásokban (beleértve az Internet Exploreren kívül más webböngészőket is). A CardSpace, az által, hogy a Windows használatához szükséges azonosítók kiválasztásának egyszerű módját nyújtja, kulcsszerepet játszik a metarendszerben, azáltal, hogy az azonosítás alapvető problémáját megoldja, nagyon fontos szerepet játszik a .NET 3.5-keretrendszerben is.

A fejlődés összefoglalása: a .NET 3.5-keretrendszer és elődei

A .NET 3.5-keretrendszer a Microsoft legfőbb fejlesztési platformja fejlődésének legutóbbi lépése, mint minden egyes korábbi, az előtte lévőre épül. A legújabb kiadás a .NET 3.0-keretrendszert foglalja magában, és nem tartalmaz radikális változásokat (breaking change). Hasonlóképpen, a .NET-3.0-keretrendszerbe beépítették a 2.0 kiadást, és ez sem tartalmazott éles változásokat. Hogy tisztán lehessen látni ennek a fejlődésnek a fokozatait, az 1.5. ábra mutatja, mivel lett több a 3.0 és a 3.5 kiadás.



1.5. ábra: *A .NET 3.5 többlete a 3.0-hoz képest*

A .NET-keretrendszerben minden a Common Language Runtime (CLR) működésén múlik. A .NET-keretrendszer osztálykönyvtáraként ismert osztályok nagy része a CLR-re épül. Ez a könyvtár minden kiadásnál bővült, ahogyan az ábrán látható. A .NET 2.0-keretrendszer nyújtotta a korszerű fejlesztői környezet alapjait, beleértve az alaposztálykönyvtárakat, az ASP.NET-et, az ADO.NET-et és a többit. A .NET 3.0-keretrendszer ezekből semmit sem változtatott, de négy új, fontos technológiát (WCF, WF, WPF és Windows CardSpace) hozzátett.

A .NET 3.5-keretrendszer változásai a 3.0 kiadás néhány részére is hatással vannak. Az ASP.NET AJAX-támogatással rendelkezik, míg a LINQ használható ADO.NET-hez vagy más módon is. Különböző kiegészítések vannak az alaposztálykönyvtárhoz, például a támogatási beállítások típusainak kiegészítése – egyedi elemek rendezetlen gyűjteménye – és a titkos támogatás tökéletesítése. A WCF, a WF és a WPF szintén mind ezt fokozzák, ahogyan később ebben az áttekintésben leírjuk. Az operációs rendszerek körének frissítése is megtörtént: a .NET 3.5-keretrendszer csak Windows Server 2008, Windows Server 2003, Windows Vista és Windows XP alatt fut.

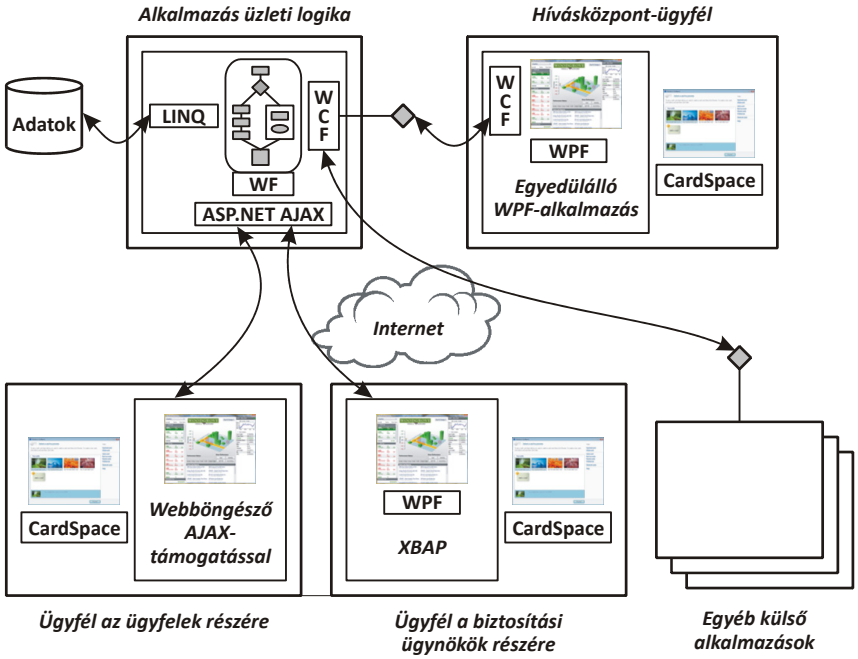
Mivel mindegyik kiadás az őt megelőzőhöz lett hozzáadva, a korábbi verziókhoz készült alkalmazások újratesztelésének szükségessége minimálisra csökkent. És mivel mind a három verzió képes egyszerre futni, még az is lehetséges, hogy alkalmazásokat a keretrendszer korábbi verziójában futtassunk, ha szükséges. A Visual Studio 2008 lehetővé teszi a keretrendszer egy különleges verzióját megcélzó projektek létrehozását. Egy ilyen módon felépített alkalmazás csak a bináris fájlokat fogja használni ehhez a verzióhoz, és a fejlesztő a Visual Studiónak és a Frameworknek csak azt az oldalát fogja látni, amelyik ebben a régi világban dolgozik. Például, az a fejlesztő, aki a .NET 2.0-keretrendszert célozza meg új alkalmazás létrehozásához, úgy látja, mintha a Frameworknek csak ez a régebbi verziója lenne elérhető.

A .NET 3.5-keretrendszer alkalmazása. Forgatókönyv

Annak, hogy megértsük, a technológiák egy csoportja hogyan dolgozik együtt az az egyik módja, ha megvizsgáljuk használhatóságuk egy példáját. Például, képzeljünk el egy alkalmazást, amely lehetővé teszi ügyfelek és ügynökök számára biztosítási ajánlatok megkötését. Ha .NET 3.5-keretrendszer használatával valósítjuk meg, akkor ez az alkalmazás úgy nézhet ki, mint azt az 1.6. ábrán láthatjuk.

Az alkalmazás programlogikáját WF-munkafolyamat használatával valósítottuk meg, ahogyan a diagram bal felső részében látható. A biztosítási ajánlatkérések kezelése többlépéses folyamat, beleértve az ajánlat kiértékelését a szervezet biztosításaival szemben, esetleg a kérvényező hitelének ellenőrzését, és talán még egy vezetői jóváhagyást is. A munkafo-

lyamat minden egyes lépését külön tevékenységként valósítjuk meg, más szükséges programokra támaszkodva. A munkafolyamat tevékenységei, a tárolt adatokhoz való hozzáférés, a LINQ lehetőségeit használja SQL-lekérdezések létrehozásához.



1.6. ábra: *Biztosítási ajánlatok megkötése .NET 3.5-keretrendszer használatával*

Ez a biztosítási cég rendelkezik hívásközponttal (call center), így lehetővé teszi ügyfelei számára, hogy biztosítást igényeljenek telefonon keresztül. A hívásközpont személyzete által működtetett ügyfélszoftvert külön WPF-alkalmazásként valósítottuk meg, ahogyan a diagram jobb felső részén látható. Ez az ügyfél a WCF-en keresztül kommunikál az alkalmazás programlogikájával, amihez a WCF-WCF kommunikációhoz optimalizált bináris protokollt használja. Ahogy az ábrán is látható, a hívásközpont dolgozói Windows CardSpace-re támaszkodva választják ki az azonosítót, amelyet használni fognak, amikor belépnek ebbe az alkalmazásba.

Az ügyfelek igényelhetnek biztosítást a weben keresztül is. Az alkalmazás, hogy ezt lehetővé tegye, a webböngészővel való kommunikáláshoz ASP.NET AJAX-ot használ, megfelelő felhasználói felületet nyújtva az ügyfelek számára. Ahogyan az 1.6. ábra bal alsó sarkában látható, az alkalmazást webböngészőn keresztül elérő ügyfél CardSpace-t használhat annak az azonosítónak a kiválasztására, amelyet szeretne megjeleníteni.

Az alkalmazáshoz interneten keresztül kapcsolódó biztosítási ügynöknek jóval gyakorlatiasabb felületre van szüksége, mint amilyet az ügyfelek látnak. Ennek megfelelően AJAX helyett inkább XBAP-felületre támaszkodhat. Ahogyan az ábra középső, lenti részén látható, ez a felhasználói felületnek a hívásközpontban használt WPF-alkalmazás által nyújtott funkcióit nagyobb részt elérhetővé teszi az ügynökök számára. Mindkettő azonos alapokra épül, és így az alkalmazás fejlesztői ugyanazokat a kódokat újra felhasználhatják mindkét ügyféltípusnál. És éppen úgy, mint a más ügyfélfajtáknál, az ügynökök CardSpace-t használhatnak az alkalmazásban megjeleníteni kívánt azonosító kiválasztására.

Végül valószínű, hogy ennek az alkalmazásnak rendelkeznie kell hozzáférési lehetőséggel, és hozzáférhetőnek kell lennie más alkalmazások által. Például, ha egy ügyfél jóváhagyásához hitelességvizsgálat szükséges, ez nagy valószínűséggel külső szolgáltatás hívásán keresztül fog megvalósulni. Lehet, hogy az alkalmazás a kérést közvetlenül egy másik alkalmazástól fogja megkapni, felfedve a szolgáltatást, amelyet ez a külső alkalmazás segítségül hívhat. Ilyen esetekben, ahogyan a jobb alsó ábrán látható, az alkalmazás WCF-et használ a szabványos webszolgáltatások segítségével végzett kommunikációhoz. A WCF támogatása a SOAP számára közvetlen kapcsolatot létesít bármelyik technológiával, amelyre ezek az alkalmazások épülnek.

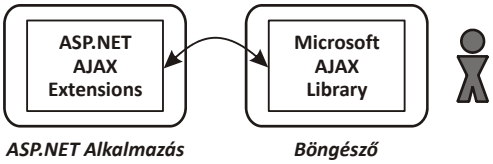
Ez a forgatókönyv illusztrálja, hogy a .NET 3.5 milyen sok komponense használható egy mintaalkalmazásban. Jó néhány lehetőség kimaradt, és így ez az egyszerű példa nem teljes illusztrációja annak, amit ez a technológiacsalád kínál. Ehelyett a cél az, hogy tiszta képet adjunk arról, hogyan használhatjuk valós üzleti problémák megoldására a .NET-keretrendszer különböző részeit együttesen.

A .NET 3.5-keretrendszer megértése: a technológiák

Hogy jobban megérezzük, mit nyújt a .NET 3.5-keretrendszer, hasznos lehet mélyebbre ásni a komponensei között. Ez a rész rövid betekintést nyújt valamennyi technológiába, amelyeket később közelebbről is megvilágítunk.

ASP.NET AJAX

Az ASP.NET manapság az egyik legnépszerűbb technológia a webes alkalmazások létrehozására. Ezeknek az alkalmazásoknak a tervezésénél az AJAX domináns szemléletté vált. Ez a két tény értelemszerűen magában foglalja, hogy az ASP.NET AJAX lesz az egyik leginkább széles körben használt része a .NET 3.5-keretrendszernek. Az 1.7. ábra bemutatja a technológia két fő összetevőjét.



1.7. ábra: *Az ASP.NET és az AJAX viszonya*

Ahogyan az ábrán is látható, az ASP.NET-alkalmazás használhatja az ASP.NET 2.0 AJAX Extensionst, amely egész sor kiszolgálóoldali vezérlőelemmel szolgál a kód és a webböngésző közötti interakcióhoz. Ezek közül a vezérlőelemek közül a legfontosabb az UpdatePanel, amely lehetővé teszi a csak a felhasználó által látható oldal egy részének frissítését, ahelyett, hogy az egészet áthelyezné. Az ASP.NET 2.0 AJAX Extensions webszolgáltatásokkal is szolgál, amelyek ügyféloldali kódból is hívhatók, hogy hozzáférjünk az alkalmazás által nyújtott információkhoz.

Az ügyfél számára az ASP.NET AJAX a Microsoft AJAX könyvtárat kínálja. Ez a könyvtár keretet hoz létre az ügyféloldali szkriptek számára, és fut Internet Explorer, Firefox, Opera és Safari alatt. Mint minden AJAX-implementáció, ez a ügyfélkód is egy webböngésző által szolgáltatott, XMLHttpRequestnek nevezett objektumot használhat. Ha ebben az objektumban meghívunk egy metódust, az egy szkriptadatot kérhet HTTP-n keresztül egy alkalmazástól, amelyet ASP.NET vagy más technológia, például a PHP, használatával hoztak létre. Ez a kérés létrehozható aszinkron módon, lehetővé téve ezzel, hogy a nélkül jussunk adatokhoz, hogy leblokkolnánk a felhasználót, míg a válaszra várunk. A visszakapott adathoz Document Object Model (DOM) használatával XML-ként juthatunk hozzá. Az XML mellett az ASP.NET AJAX lehetővé teszi azt is, hogy JavaScript Objet Notation (JSON) segítségével reprezentált adatokat cseréljünk.

Az AJAX-típusú webes alkalmazás, amikor az elképzelés először felmerült 1999-ben, jó ötlet volt. Nem hiába vártunk éveket arra, hogy ez a szemlélet bekerüljön a fővonalba, az AJAX-alkalmazások ma normává váltak. A Microsoft döntése, hogy az ASP.NET AJAX-alkalmazást belefoglalja a .NET 3.5-keretrendszerbe, biztos, hogy segített ennek a hasznos szemléletnek a gyors elfogadásában.

Language-Integrated Query (nyelvbe ágyazott lekérdezés)

Közös szemlélet létrehozása különböző adatok elérésére, nem könnyű feladat. Ezt a szemléletet a fejlesztők számára érthetővé – nem pedig még bonyolultabbá – tenni még nehezebb. A LINQ az adatokkal való munkához egy általános és eléggé ismert szintaxtist használ ennek megoldására.

Íme egy egyszerű LINQ-lekérdezés C#-ban, amely adatokat olvas:

```
var exampleQuery =
      from s in Students
      where s.Gender == "M"
      select s.Name
```

Ez a lekérdezés a férfi diákok nevét adja vissza egy sztringlista formájában az exampleQuery változóban tárolva. A lista kiíratásához a program a következőket teszi:

```
foreach (string name in exampleQuery) {
      Console.WriteLine(name);
}
```

A lekérdezés szintaxisa SQL-re emlékeztet, amely ma az irányadó nyelv relációs adatok eléréséhez. Ez érthető, mivel az SQL széles körben használatos, sok fejlesztő ismeri. De azért fontos annak megértése, hogy a fent bemutatott LINQ-lekérdezés, bár úgy néz ki, mint az SQL, nem beágyazott SQL-utasítás, hanem tiszta C#, magának a nyelvnek a része. Ez azt jelenti, hogy a lekérdezés más programváltozókat használhat, elérhető a hibakereső program számára, és a többi. A technológiának azért az a neve, hogy „Language-Integrated Query" (nyelvbe ágyazott lekérdezés), mert a különböző típusú adatok lekérdezésének utasításait közvetlenül a programnyelvbe építjük be.

Az SQL-hez való hasonlósága ellenére ez a lekérdezés példa csak relációs adatok hozzáférhetőségére korlátozódik. Tulajdonképpen a .NET 3.5-keretrendszer több különböző LINQ-változatot foglal magában, amelyek mind azonos alapszintaxist használnak a lekérdezéshez. Ezek a változatok a következők:

- LINQ to ADO.NET: objektum/relációs (O/R) leképezést végez. A LINQ-nek ez a területe két lehetőséget foglal magában. Az első, melyet úgy nevezhetünk, hogy „LINQ to SQL": ez a fentihez hasonló lekérdezést fordít le SQL-lekérdezéssé, majd táblázatosan megjeleníti az SQL Server adatbázisban. A második opció, az úgymond „LINQ to DataSet": ez egy ADO.NET által visszaadott adathalmaz tartalmán hajt végre a fentihez hasonló lekérdezést.
- „LINQ to Objects": lehetővé teszi memórián belüli adatszerkezetek, mint például az objektumhierarchiák gyűjteményének lekérdezését. A fenti lekérdezés származhat ebből az adatfajtából is, az SQL-hez való hasonlósága ellenére ez nem korlátozódik csak a táblázatban található adatok elérésére.
- „LINQ to XML": lehetővé teszi XML-adatok lekérdezését. Ez a szemlélet egy kicsit más szintaxist használ, kifejezve az XML és a programnyelv világa közötti leképezés egyedi szükségletét, de a LINQ-lekérdezés alapszerkezete azonos marad.

Az SQL-hez hasonlóan a LINQ is definiál további operátorokat a lekérdezésekhez. Ebbe beletartoznak olyan dolgok is, mint például az OrderBy, amely meghatározza, hogyan rendezzük az eredményeket; a GroupBy, amely a kiválasztott adatot csoportokba szervezi; és az aritmetikai operátorok, mint például a Sum. És még egyszer, ezek általában LINQ-változókon keresztül használhatók, nem csak „a LINQ az SQL-hez" opcióhoz vannak kötve.

A LINQ létrehozói különböző célokat irányoztak elő, többek között az O/R leképezést a .NET-alkalmazásokhoz, közös szintaxis lehetőségét a különböző típusú adatokkal való munkához, a szintaxis integrálását közvetlenül a programnyelvbe, és egyebeket. Ahogy már annyiféleképpen leírtuk ebben a bevezetőben, a cél az, hogy megkönnyítsük a fejlesztők életét, akik Visual Studio 2008-at és .NET FramWork 3.5-öt használnak munkájukhoz.

Windows Communication Foundation

A szolgáltatás-orientált kommunikációra történő váltás azt jelenti, hogy megváltozik az alkalmazások együttműködésének módja. A szolgáltatásorientált alkalmazások támogatásához tervezett WCF határozottan ezt a változtatást tükrözi. Ez a rész leírja a WCF legfontosabb szempontjait, beleértve a kiszolgálókat és az ügyfeleket, a kommunikációs lehetőségeket, a biztonsági támogatásokat, a megbízható kommunikációt és a tranzakciókat.

Kiszolgálók és ügyfelek

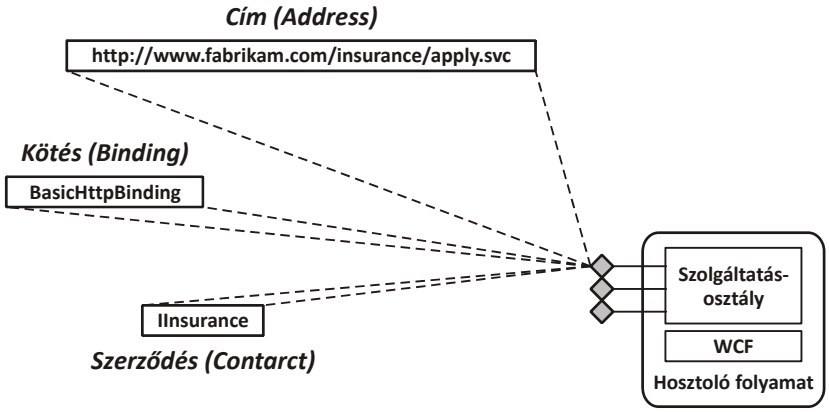
A WCF alapötlete egyszerű: a kiszolgáló megmutat egy felületet, amely az ügyfél számára hozzáférhető. Ezt a felületet definiálhatjuk webszolgáltatás-leírónyelv (Web Services Description Language – WSDL) használatával, majd kódba fordításával, vagy közvetlenül a nyelvben definiálhatjuk, például C#-ban vagy Visual Basicben. Egy biztosítási ajánlati szolgáltatásokat megjelenítő egyszerű felület az utóbbi megközelítés esetén így alakul:

```
[ServiceContract]
interface IInsurance
{
  [OperationContract]
  int Submit(int policyType, string ApplicantName);

  [OperationContract]
  bool CheckStatus(int applicationNumber);

  [OperationContract]
  bool Cancel(int applicationNumber);
}
```

Ennek a C#-felületnek a definícióját a `ServiceContract` tulajdonsággal adjuk meg. Ez az attribútum mutatja, hogy a WCF ezen a felületen távolról lehívható műveletek formájában képes metódusok megjelenítésére. Az, hogy melyik interfészmetódus látható, attól függ, hogy melyiknek adtunk meg `OperationContract` tulajdonságot. Ebben az egyszerű példában minden metódust ezzel a tulajdonsággal jellemzünk, és így mindegyikük látható lesz távoli hívások számára. Ez nem szükséges, teljesen megfelelő csak egy bizonyos interfészmetódushoz használni az `OperationContract` tulajdonságot. Bármelyiket választjuk is, az interfészt az alkalmazás valamely osztályának úgy kell létrehoznia, hogy gondoskodik aktuális kódról a felület által definiált metódus számára.

Ha ezt megtettük, a WCF automatikusan létrehozza az `OperationContract` tulajdonsággal jellemzett metódusokat, amelyek a szolgáltatás ügyfelei számára hozzáférhetőek.

Az 1.8. ábra valamivel több részletet mutat meg abból, hogy a szolgáltatás éppen hogyan mutatja meg magát a ügyfélnek. Az ügyfél, a helyett, hogy egy felületet közvetlenül érne el, inkább egy meghatározott végponthoz kapcsolódik. Egy szolgáltatás több végpontot tud elérhetővé tenni, lehetővé téve több különböző ügyfél egyidejű kapcsolódását különböző módokon.



1.8. ábra: *A szolgáltatás ügyfélkapcsolatai*

Ahogyan az ábrán látható, minden egyes végpont három dolgot határoz meg:

- Egy *címet (address)*, ami jelzi, hogy hol található a végpont. Ahogyan az ábrán is látható, a címet URL-ben fejezzük ki.
- Egy *kötést (binding)*, ami leírja, hogyan lehet a végpontot használni. Minden kötés jellemez néhány dolgot, például, hogy milyen protokollt használ a műveletek végrehajtásához, milyen típusú biztonságot alkalmaz és így tovább. A WCF sok előre definiált kötést tartalmaz a legtöbb általános esethez, mint például az itt bemutatott BasicHttpBinding, és lehetőség van egyedi kötések definiálására is. Mivel egy szolgáltatás több végpontot tud elérhetővé tenni, mindet különböző kötéssel, lehetővé tud tenni egyszerre több hozzáférést különböző típusú ügyfelek számára eltérő protokollokon keresztül és eltérő biztonsági beállításokkal.

- Egy *szerződést (contract)*, amely leírja a műveleteket, amelyek a végponton keresztül végrehajthatók. A szerződés azonosítható csak az interfész nevével, amely meghatározza azokat a műveleteket, amelyek itt biztosítottak.

A WCF alapjai egyszerűek. Ahogy a legtöbb kommunikációs technológia esetében, a részletezés bonyolult lehet – rengeteg lehetőség van –, de átlagos WCF-alkalmazás létrehozása nem nehéz.

REST- és RSS/ATOM támogatása

A .NET 3.0-keretrendszer eredeti kiadásában a WCF webszolgáltatástámogatása a SOAP-ra összpontosult. Néhány esetben azonban a webszolgáltatásokhoz más szemlélet megfelelőbb lehet. A Representational State Transfer (REST)-ként ismert szolgáltatás közvetlenül a web technológiájába van beépítve. A REST ahelyett, hogy új protokollt definiálna, ahogy a SOAP alkotói tették, a HTTP olyan beépített műveleteit használja, mint a GET, a POST és mások. És ahelyett, hogy az elért információkat XML-ben definiált paraméterekkel azonosítaná, ahogyan a SOAP tipikusan teszi, a REST azt feltételezi, hogy minden URL-lel azonosított.

Ennek a szemléletnek a támogatására a WCF a .NET 3.5-keretrendszerben új kötéssel bővült, ez a WebHttpBinding, amely az információt közvetlenül HTTP-n keresztül küldi. Két új tulajdonság is létrejött, ahogy ez a példa mutatja:

```
[ServiceContract]
interface IAccount
{
  [OperationContract]
  [WebGet]
  int GetBalance(string account);

  [OperationContract]
  [WebInvoke]
  int UpdateB
}
```

Ez az egyszerű felület egy RESTful-szolgáltatást tesz elérhetővé, olvasáshoz és banki követelések frissítéshez. Annak jelzésére, hogy a GetBalance hívás továbbítása közvetlenül a HTTP-n keresztül történik, a műveletet egy WebGet tulajdonsággal jellemezzük. Hasonlóképpen, ha az UpdateBalance művelet a WebInvoke tulajdonságot kapja, a HTTP POST-on ke-

resztül továbbítódik. (Megadhatjuk azt is, hogy az ezzel a tulajdonsággal felruházott művelet valamely másik HTTP-műveletet használjon, például a PUT-ot vagy a DELETE-et.) Ez az egyszerűbb, a SOAP- és a WS-* protokollok használatát helyettesítő RESTful-stílus hozzáadódik a webalapú kommunikációk hagyományos alapelemeihez. Mivel mindegyik szemléletnek van létjogosultsága valamelyik helyzetben, a WCF a .NET 3.5-keretrendszerben mindkettőt támogatja.

Az RSS és az ATOM által definiált XML-formátumok küldésének képessége sokkal jelentősebb lett, mióta a WCF megjelent. Következésképpen a .NET-keretrendszer 3.5-ös verziójában található WCF beépített támogatásokat nyújt ehhez a két formátumhoz. Bármelyik strukturáltan használt információt küldhetjük bármely WCF-kötésen keresztül, bár valószínűleg az új WebHttpBinding lesz a leggyakoribb választás. A WCF-ellátású formázókat használó .NET 3.5-keretrendszer alkalmazások létre tudnak hozni, vagy fel tudnak használni RSS- vagy ATOM-táplálást úgy, hogy egyszerűen ezzel az egyre nagyobb jelentőségű tartalommal dolgoznak.

Kommunikációs lehetőségek

A különböző fejlesztők által felépített különböző alkalmazások más-más módon kommunikálnak. A legtöbb fejlesztő számára a legegyszerűbb a távoli eljáráshívás (RPC), amellyel az ügyfél inkább távoli műveleteket hív meg, mint helyieket. Például ahhoz, hogy a korábban mutatott felületet kapjuk, az ügyfél bármely műveletet hívhatja a szokásos szinkrón módon, türelmesen várva, míg a válasz megérkezik. Ez a lehetőség egyszerű a fejlesztő számára, és bizonyos környezetben ez a jó választás.

A WCF néhány más lehetőséget is nyújt. Ezek a következők:

- Hívások, amelyekre nincs válasz. Az ilyen típusú kommunikáció `OneWay` tulajdonsággal felruházva hasznos lehet események küldésénél vagy más egyirányú interakcióknál.
- Aszinkron üzenetalapú kommunikáció MSMQ-n keresztüli közvetlen küldésekre vagy fogadásokra használhatók.
- SOAP-üzenetek explicit kezelése, beleértve az elemek beszúrásának képességét közvetlenül a SOAP fejébe.

A WCF lehetővé teszi a fejlesztők számára annak különböző szempontok szerinti ellenőrzését, hogyan viselkedik a szolgáltatás. Például, a `ServiceBehavior` tulajdonság használatával be tudjuk állítani, hogy a szolgáltatás egy- vagy több szálon fusson, hogy a szolgáltatás egy új példánya minden egyes híváskor létrejöjjön stb.

Biztonság, megbízhatóság és tranzakciók

Az alapkommunikáció – adatok mozgatása rendszerek között – nagyon hasznos, de ritkán elég. A legtöbb alkalmazásnak többre van szüksége. Például, az elosztott alkalmazások nagy többségének szüksége van a biztonság valamilyen formájára. A biztonság bonyolult dolog, napjainkban rengeteg különböző szemlélet és technológiák sokasága használatos. A WCF elsődlegesen a kötéseket használja a biztonság megteremtéséhez, így a fejlesztők számára lehetővé teszi biztonságos, elosztott alkalmazások létrehozását a nélkül, hogy rájuk erőltetné minden részlet megértését. Például, a korábban bemutatott BasicHttpBinding inkább HTTPS használatához alkalmas, mint a sima HTTP-hez, és más kötések is több biztonsági beállítással szolgálnak. Például, a WsHttpBinding támogatja a WS-Securityt, lehetővé téve az együttműködő SOAP-alapú hitelesítést, adatintegritást és adatbizalmasságot. A fejlesztő létrehozhat egyedi kötést is, amely pontosan azokat a biztonsági szolgáltatásokat nyújtja, amelyekre az alkalmazásának szüksége van.

Sok alkalmazás számára nélkülözhetetlen, hogy megbizonyosodjunk a kommunikáció megbízhatóságáról. A hagyományos webszolgáltatás, SOAP küldése HTTP-n keresztül néhány esetben kielégítő, éppen ez történik, ha a BasicHttpBinding kötést használjuk. Rengeteg olyan helyzet van azonban, amikor ez a széles körűen használt választás nem elég. Például, az üzenet, amely keresztülmegy egy vagy több SOAP-közvetítőn, nem elégedhet meg ezzel az egyszerű szemlélettel a vég-vég típusú kommunikáció megbízhatósága tekintetében. Ezekben az esetekben a WCF a WS-ReliableMessaginget hajtja végre. Ha a fejlesztő olyan kötést választ, amely támogatja ezt az opciót, mint például a WsHttpBinding, akkor automatikusan együttműködő megbízható üzenettovábbítást kap.

Az elosztott tranzakciók is fontosak lehetnek néhány alkalmazás esetében. A WCF a System.Transactionsra épít, amely eredetileg a .NET-keretrendszer 2.0 kiadásában jelent meg, és ezzel teszi lehetővé tranzakciókat szolgáló programok létrehozását. Egy metódus használhatja az `Operation-`

`Behavior` tulajdonságot annak jelzésére, hogy tranzakciót kér, és annak meghatározására, hogyan viselkedjen a tranzakció. A WCF a WS-AtomicTransaction specifikációt használja szállító határokkal együttműködő, elosztott tranzakciók engedélyezéséhez. Az ezekben a többszállítós megállapodásokban definiált technológiák használata révén a WCF-alkalmazások különböző technológiákon átívelő tranzakciókban vehetnek részt.

Eszköztámogatás

A jó eszközök minden fejlesztő életét megkönnyítik. Ennek megfelelően a .NET 3.5-keretrendszerben található WCF-verzió Visual Studio 2008 támogatással rendelkezik a következőkhöz:

- Projekttípusok, amelyek segítenek a fejlesztőknek nekikezdeni WCF-alkalmazások létrehozásához. Ezek a projektek a következők:
  - website (webhely) vagy Web application (webalkalmazás), amely egy WCF-szolgáltatást hosztol;
  - könyvtár megvalósítása WCF-szolgáltatásokból;
  - WCF-alkalmazás, amely megmutatja a szindikációs táplálást (syndication feed) RSS vagy ATOM használatával;
  - WCF-szolgáltatás, amelyet AJAX-ügyfél használatára terveztek. Ennek a szolgáltatásnak automatikus beállításai vannak a WebHttpBinding használatához JavaScript Objet Notation (JSON)-nal, ez egyike a kódolási lehetőségeknek, amelyeket ez a kötés nyújt.
- WCF Autohost, amely képes automatikusan hosztolni egy könyvtáralapú WCF-szolgáltatást.
- A szolgáltatáskonfiguráció-szerkesztő (Service Configuration Editor) olyan eszköz, amely egyszerűbbé teszi a WCF-konfigurációs fájlok létrehozását és módosítását.

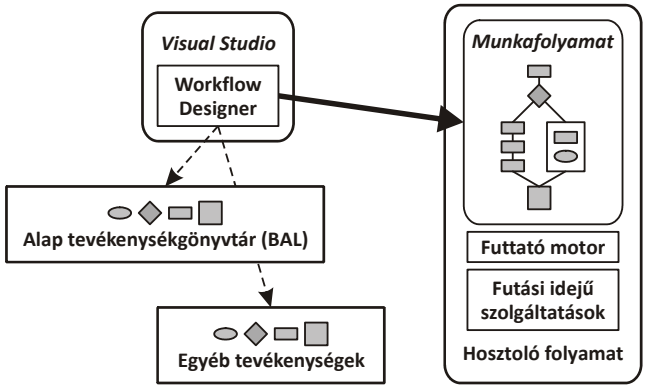
Windows Communication Foundation és más Microsoft-technológiák

Ahogyan korábban már említettük, a WCF helyettesít néhány korábbi Microsoft-technológiát az elosztott alkalmazások létrehozása terén. A legtöbb alkalmazást, amelyet eddig ASP.NET Web Services, .NET Remoting, Enterprise Services vagy WSE alkalmazásával építettünk fel, ezek helyett WCF

használatával hozhatjuk létre. A WCF-alkalmazások együtt tudnak működni ASP.NET Web Services alkalmazásokkal – mindkettő támogatja a szabvány SOAP-ot – éppen úgy, mint az Enterprise Services, az MSMQ és a WSE 3.0 verzióval létrehozott alkalmazások. Bár az új .NET 3.5-keretrendszer alkalmazásai általában nem használják ezeket. Minden technológia, amelyet a WCF helyettesít, még része a keretrendszer jelenlegi verziójának, és általában támogatást is kapnak. Az e technológiák korábbi verzióival készített alkalmazások továbbra is normálisan futnak; a .NET 3.5 Framework telepítése és használata nem teszi tönkre a létező kódokat.

Windows Workflow Foundation

A Windows-programok jelentős részénél a munkafolyamatok által vezérelt folyamatorientált tervezés lehet a megfelelő szemlélet. A WF célja az, hogy lehetővé tegyék a fejlesztők számára ilyen munkafolyamat-alapú alkalmazások tervezését és végrehajtását. Az 1.9. ábra a WF által ehhez nyújtott összetevőket mutatja.



1.9. ábra: *A WF-összetevők a munkafolyamat-alapú tervezéshez és végrehajtáshoz*

Ahogy korábban leírtuk, minden munkafolyamat tevékenységekből épül fel. A munkafolyamatok és tevékenységek osztályokat alkotnak, így mindkettő létrehozható közvetlenül a forráskódban. A WF-ben elérhető a Workflow Designer (munkafolyamat-tervező), egy Visual Studio által hosztolt grafikus eszköz munkafolyamatok készítéséhez. Bár új munkafolyamatot hozunk létre, a tevékenységeit a WF vagy más forrás által szolgáltatott alaptevékenységek könyvtárából (BAL, Base Activity Library) merítjük.

Egy már létrehozott munkafolyamatot majd a WF futtató motorja hajtja végre. Ez a motor a futásidejű szolgáltatások azon csoportjára támaszkodik, amelyek a munkafolyamat állapotát fenntartják, végrehajtását nyomon követik, és a többi. Mindezek – a futásidejű szolgáltatás, a futtatómotor és maga a munkafolyamat – néhány hosztfolyamatban tárolódik. Ez a folyamat bármilyen Windows-folyamat lehet, egy egyszerű konzoltól vagy egy asztalon futó WPF-alkalmazástól kezdve egészen a méretezhető kiszolgálófolyamatig.

Ahhoz, hogy a WF-et megértsük, legalább egy kicsit ismernünk kell minden összetevőjét. A következő rész rövid bepillantást ad ezekbe.

Munkafolyamatok

A munkafolyamat lényegében nem több, mint tevékenységek egy csoportja. A WF a beépített munkafolyamatok két típusához nyújt támogatást:

- Szekvenciális munkafolyamat: meghatározott sorrendben hajt végre tevékenységeket. Hasonlóan egy hagyományos folyamatábrához. A szekvenciális munkafolyamat tartalmazhat elágazásokat, ciklusokat és más vezérlőstruktúrákat. Alapesetben a tevékenységek lineárisan hajtódnak végre, egyik a másik után.
- Állapotgép-munkafolyamat: hagyományos állapotgépet hajt végre. Mint bármelyik állapotgépet, amely tevékenységet hajt végre egy bizonyos időben, az aktuális állapot és valamely fogadott esemény kombinációja definiálja.

A szekvenciális opció olyan jól definiált munkafolyamatok esetében hasznos, amelyeket tisztán szoftveralapú folyamatoknál használunk. Ezeket viszonylag egyszerű létrehozni és megérteni, és kezdetben sokkal természetesebbnek hatnak a legtöbb fejlesztő számára. Az állapotgép-munkafolyamat akkor jobb választás, ha a végrehajtás útja kevésbé látható előre. Jó példa erre egy olyan munkafolyamat, amely emberek közötti interakciókat hoz létre, bármelyikük meg tudja szakítani a munkafolyamatot bármely ponton. Ennek a helyzetnek a megoldása szekvenciális munkafolyamattal lehetséges, de minden egyes lépés elágazás kell, hogy legyen: „tedd ezt, ha a munkafolyamatot nem szakították meg, tegyél valami mást, ha megszakították”. Az ilyen viselkedés modellezése állapotgép használatával lényegesen egyszerűbb, mivel a munkafolyamat megszakítására irányuló kérés egyszerűen csak egy másik esemény, amelyet fogadhatunk, és kezelhetünk bármely ponton.

Az állapotgép-munkafolyamatok támogatása csak egy példája annak, hogyan próbál a WF támogatást nyújtani az emberek, valamint a rendszerfolyamatok számára. Másik példa arra, hogyan támogatja a WF a futó folyamatok változtatását: az ember időnként szeszélyes, és nem szokatlan, hogy valaki belekever a munkafolyamatba, mert szeretne egy lépést hozzáadni, törölni egy lépést vagy valami más változtatást eszközölni a folyamatban menetközben. Azoknak a fejlesztőknek, akik létrehoznak egy munkafolyamatot, hogy ezt ellenőrzött módon vigyék véghez, a WF lehetőséget ad arra, hogy meghatározzák, hogyan módosítható az a folyamat végrehajtás közben.

Az alaptevékenység könyvtár (BAL)

A fejlesztők szabadon hozhatnak létre egyedi tevékenységeket. Tulajdonképpen a Microsoft célja az, hogy elősegítse egy újrahasználható tevékenységekkel teli WF-ökoszisztéma fejlődését. Mégis, mindenki számára egyszerűbbé teszi az életet, ha az alaptevékenységek egy átlagos készletével kezdi a fejlesztést. A Base Activity Library (BAL) feladata, hogy ezt az általános készletet rendelkezésre bocsássa.

A munkafolyamatnak nem szükséges bármit is használnia a BAL-ból. Sok fejlesztő mégis úgy fogja találni, hogy a BAL, különösen kezdetben, egyszerűbbé teszi életét. A BAL-ban tárolt tevékenységek között a következők találhatók:

- IfElse: egy adott feltétel teljesülése esetén végrehajtja a két vagy több lehetséges útvonalba foglalt tevékenységeket.
- While: ismételten végrehajt egy vagy több tevékenységet mindaddig, amíg a feltétel igaz.
- Sequence: egyesével hajtja végre tevékenységek egy csoportját, meghatározott sorrendben.
- Parallel: párhuzamosan hajt végre két vagy több tevékenységcsoportot.
- Code: végrehajt egy adott kódrészletet.
- Listen: várakozik egy adott eseményre, majd végrehajt egy vagy több tevékenységet, ha az esemény megtörtént.

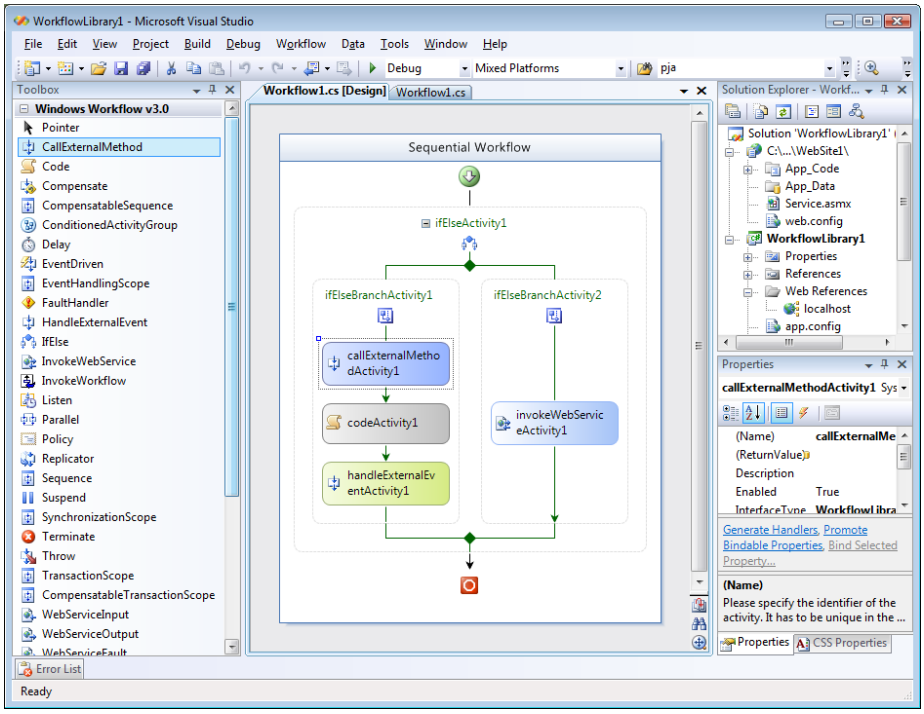
- InvokeWebService: ASP.NET Web Services használatával meghív egy webes szolgáltatást.
- State: jellemez egy állapotot egy munkafolyamat-állapotgépében.
- EventDriven: meghatároz egy olyan átmenetet, amely egy vagy több tevékenységet tartalmaz, amelyeket akkor kell végrehajtani, amikor egy adott esemény bekövetkezik egy adott állapotban.
- Policy: megengedi üzleti szabályok végrehajtását a WF támogatta Rules Engine használatával.

A WF, ahelyett, hogy egyéni nyelvet definiálna munkafolyamatok meghatározásához, sokkal általánosabb szemléletet alkalmaz a tevékenységek használatához. A BAL elérhetővé tesz egy „nyelvet" , de mindenki, aki a WF-et használja, szabadon definiálhatja a sajátját.

Eszközök a Windows Workflow Foundation használatához: a munkafolyamat-tervező

Egyik előnye annak, ha munkafolyamatok használatával hozunk létre alkalmazásokat az, hogy a munkafolyamatokat grafikusan tudjuk definiálni. Ezt a WF Workflow Designer teszi lehetővé, ahogy az 1.10. ábra is mutatja. Alaphelyzetben a BAL-ban található tevékenységek megjelennek az eszköztárban, így lehetővé teszik a fejlesztő számára, hogy behúzzák azokat az eszköz tervezőfelületére egy munkafolyamat létrehozásához.

Néhány fejlesztő jobban szeret kódot írni, nem szereti a grafikus tervezőt. A WF lehetővé teszi ezt is (és néha szükség is van erre: a tevékenységek általában közvetlenül a kódba vannak beépítve). Lehetséges a két szemlélet kombinálása is, munkafolyamat létrehozása a Workflow Designer és a közvetlen kódolás használatával. A cél az, hogy a fejlesztők azt a szemléletet használhassák, amely a leginkább hatékony számukra. A tágabb eszköztámogatás érdekében a munkafolyamatokat XAML-ben is kifejezhetjük, amely azonos a WPF által használt nyelvvel. Tulajdonképpen a Workflow Designer használatával létrehozott munkafolyamatok alapértelmezettek egy XAML-definíció számára.



1.10. ábra: *Munkafolyamat-tervezés grafikus támogatással*

A futtatómotor és a futásidejű szolgáltatások

Ahogy korábban leírtuk, a WF-futtatómotor feladata a tevékenységek végrehajtása a munkafolyamatban. Ennek részeként a futásidejű szolgáltatások egy csoportjára támaszkodik. A WF magában foglalja ezeknek a szolgáltatásoknak szabványos implementációját, de ambiciózus fejlesztők igényeinek megfelelően helyettesíthetik azokat. Ezek a szolgáltatások több különböző dolgot támogatnak, de kettő nagyon fontos:

- Állandóság: egy munkafolyamat, amely valamely eseményre várakozik, használhatja ezt a szolgáltatást a memórián belüli állapotának lemezre történő automatikus mentéséhez. Ha az esemény bekövetkezik, a szolgáltatás automatikusan visszaolvassa a munkafolyamat állapotát, és újraindítja a végrehajtást. Ez különösen hasznos olyan munkafolyamatoknál, amelyek embereket vonnak be, mivel órák, napok vagy még több idő telhet el, míg a válaszra várnak.

- Nyomkövetés: a tevékenységek egy munkafolyamatban tisztán körülhatárolják a folyamat végrehajtását, amelyet megvalósítanak. A WF nyomkövetési szolgáltatása lehetővé teszi a fejlesztők számára információk létrehozását a munkafolyamat végrehajtásáról, amely automatikusan az adatbázisba íródik. Például, egy fejlesztő szeretné nyomon követni azt, amikor a munkafolyamat indul és befejeződik, amikor az egyes tevékenységek kezdődnek és véget érnek, valamint más információkat is.

Munkafolyamat-engedélyezett szolgáltatások

A WF és a WCF nyilvánvaló kombinációs lehetőséget kínálnak. A munkafolyamatoknak gyakran szükségük lesz szolgáltatások segítségül hívására, és sokszor hasznos lehet, ha egy szolgáltatást egy munkafolyamattal valósítunk meg. A .NET-keretrendszer 3.0-ás verziójában lévő első WF-kiadásban nem volt könnyű ennek a két technológiának a kombinálása, de a helyzet sokat fejlődött a .NET-keretrendszer 3.5-ös verziójában. Ebben a verzióban könnyen együtt lehet használni a WF-t és a WCF-t a munkafolyamat-engedélyezett szolgáltatások létrehozásához.

Ez a kombináció két új WF-tevékenységtől függ:

- Küldés: kérést küld WCF használatával, majd válaszra vár. A fejlesztő megadja a műveletet, amelyet segítségül kíván hívni, és azt a végpontot, amelynél ez a művelet megtalálható.
- Fogadás: a WCF révén fogad egy bejövő kérést, majd küld egy választ. A fejlesztő csak annak a műveletnek a nevét adja meg, amely elfogadja ezt a bejövő kérést.

Ez a két tevékenység ugyanúgy behúzható egy WF-munkafolyamatba, mint bármilyen másik, ezután igény szerint konfigurálható. A cél az, hogy hasznos módon lehetővé tegyék a WF és a WCF kombinálását.

Windows Workflow Foundation és más Microsoft-technológiák

Egy új szemlélet bevezetése elkerülhetetlenül hatással van a már létezőkre. A WF esetében ez a hatás még erősebben érződött a Windows SharePoint Servicesnél, a Microsoft Office 2007 rendszernél és a BizTalk Servernél.

A Windows SharePoint Services 3 verziója a WF-et hosztolja futásidőben, hogy a fejlesztők könnyebben hozhassanak létre munkafolyamat-alkalmazásokat dokumentumok együttműködéséhez és az információk megosztásának egyéb fajtáihoz. Az Office SharePoint Server 2007, amely az Office 2007 része, a WF támogatására épít a Windows SharePoint Servicesben. Egyéb dolgok mellett ennek a kiszolgálónak a hozzáadása lehetővé teszi az InfoPath űrlapok megjelenítését közvetlenül az Office 2007 ügyfélalkalmazásban és az általános forgatókönyvekhez előre definiált munkafolyamatok használatát, mint például a dokumentumok jóváhagyása.

Aki jártas a BizTalk Server használatában, biztosan észrevette mostanra a hasonlóságot ennek a terméknek a vezénylési lehetőségei és a között, amit a WF nyújt. Tulajdonképpen a BizTalk Server 2006 R2-t követő nagyobb kiadást arra tervezték, hogy helyettesítse a termék létező vezénylési funkcióit a WF-fel úgy, hogy eszközöket nyújt a létező vezénylések átviteléhez a WF-folyamatokba.

Mivel ez a Windows szabványos munkafolyamat-technológiája, a WF meg fog jelenni más Microsoft-termékekben és -technológiákban is. A WF otthonra lelt számos alkalmazási szoftverfejlesztő (ISV) és cég által létrehozott alkalmazásban is. Mivel nem minden Windows-alkalmazás épül fel munkafolyamatként, a WF megkönnyíti a fejlesztők életét azokkal kapcsolatban, amelyek így épülnek fel.

Windows Presentation Foundation

Mind a szolgáltatásorientált kommunikáció, mind a munkafolyamat-alapú logika fontos a korszerű alkalmazásokban. A felhasználók gyakran többet törődnek azzal, amit látnak: a felhasználói felülettel. A WPF célja, hogy megfeleljen a korszerű alkalmazásokhoz létrehozott felhasználói felületek kihívásának. Ahogy a következőkben leírjuk, a WPF a lehetőségek széles körével rendelkezik ehhez.

A Windows Presentation Foundation lehetőségei

Egy fejlesztő szabadon hozhatja létre egy WPF-alkalmazás felületét tisztán C#, Visual Basic vagy más CLR-alapú nyelv használatával. Ahogy korábban leírtuk, a WPF lehetővé teszi egy felület megadását XML-alapú XAML használatával is. Az elemek és a tulajdonságok az XAML-ben közvetlenül a WPF által nyújtott tulajdonságokhoz és osztályokhoz kapcsolódnak. Például, itt egy egyszerű XAML-ban definiált gomb:

```
<Button Background="Red“>
  No
<Button Background="Red“>
```

Ez a példa egy piros gombot hoz létre, amely a „No” szöveget tartalmazza. Pontosan ugyanezt az eredményt kaphatjuk a következő kóddal:

```
Button btn = new Button();
btn.Background = Brushes.Red;
btn.Content = “No";
```

Bár ez így van definiálva, egy WPF-alkalmazás támaszkodhat az alapelrendezés paneljeire. Minden panel tipikusan tartalmaz vezérlőket, beleértve a Buttont, a Textboxot, a ComboBoxot, a Menut és a többit, amelyeket a WPF révén tud szolgáltatni. Ezeknek a vezérlőelemeknek az elhelyezkedése függ attól, hogy milyen típusú panelt választunk. Egy Grid például, lehetővé teszi a vezérlőelemek elhelyezését meghatározott rácson, míg a Canvas megengedi a fejlesztőnek a vezérlőelemek elhelyezését bárhol a határain belül. És ahogy az a GUI-ban szokásos, a felhasználó által kiváltott események elkapása és kezelése különböző vezérlőelemekkel és más alkalmazásbeli osztályok használatával történik. Lehetséges stílusok és sablonok használata is vezérlőelemek csoportjához, amely az alkalmazás számára könnyebbé teszi egységes kinézet létrehozását.

A WPF jóval többet támogat, mint ezek az alapfelhasználói felületi funkciók, beleértve a következőket:

- Dokumentumok: a WPF-alkalmazás a XAML FixedDocument címkéjének használatával meg tud jeleníteni egy XPS-dokumentumot. Az alkalmazás dokumentumfolyamatokat is meg tud jeleníteni a FlowDocument címke segítségével. Egy dokumentumfolyamat úgy tud viselkedni, mint egy hagyományos képernyőn megjelenített dokumentum, lehetővé téve ezzel a felhasználó számára a tartalom végiggörgetését. Különböző tulajdonságok beállításával ezen a címkén a fejlesztő a dokumentumot a környezetéhez alkalmazkodóbbá tudja tenni. Például, a dokumentum oldalanként tud megjelenni, megszabadítva az olvasót attól, hogy visszafelé vagy előre kelljen görgetnie. A cél az, hogy, amennyire lehet, lehetővé váljék képernyőn megjelenített dokumentumok olvashatóvá tétele.

- Grafikák: a WPF-támogatást foglal magában kétdimenziós és háromdimenziós vektorgrafikák létrehozásához. 2D munkához a WPF szabványos absztrakciókat nyújt, mint például alakzat, ecset és toll, míg 3D grafikához lehetővé teszi egy modell definiálását, amely megvilágítást és kamerapozíció-információkat tud meghatározni. A korábbi technológiáktól eltérően, mint például a Windows Forms, amely GDI+-ra támaszkodik a grafikákhoz, a WPF-grafikák nem zárkóznak el fogalmak független csoportjának használatával, amelyeket a fejlesztőknek meg kell érteniük. Ehelyett a grafikákhoz használt XAML-elemek természetesen kombinálhatók azokkal, amelyek bármely máshoz használatosak a felhasználói felületen.
- Képek: XAML Image címke használatával a WPF-alkalmazás képes megjeleníteni különböző formátumú képeket, mint például JPEG, GIF és mások.
- Média: egy WPF-alkalmazás tudja használni a MediaElement címkét különböző formátumú videók és audiók megjelenítésére, mint például WMV, AVI és MPEG.
- Animáció: a WPF rendelkezik beépített támogatással animációk használatához, amelyek a felhasználói felületek legtöbb részén alkalmazhatók. Például, egy kör nőhet és összehúzódhat, egy gomb simán változtathatja a méretét. Az alkalmazások meghatározhatnak idővonalat tartalmazó diavetítéseket is, lehetővé téve a bekövetkező animációk sorrendjének ellenőrzését.
- Adatkötés: mivel sok WPF-alkalmazás adatokat jelenít meg, hasznos, ha automatikus támogatással rendelkezünk az adatok felhasználói felületek elemeihez történő rendeléséhez. A WPF rendelkezik az adatkötés ezen fajtájával az objektumokban tárolt információk számára és más forrásokhoz.

A Windows Presentation Foundation alkalmazása

A WPF a felhasználói felület funkcióinak széles skáláját nyújtja, lehetővé téve a fejlesztők és a tervezők számára minél vonzóbb felhasználói felületek létrehozását. Bár az mindegy, hogy egy ügyfélalkalmazás milyen csinos, néhány szervezet tiltakozhat a használata ellen a telepítés miatt. Ha egy ügyfél új verziója fizikai kapcsolatot igényel minden egyes asztali gépnél, amelyre ezt az alkalmazást installálják, akkor a frissítések bekerü-

lési költsége jelentős lehet. A problémát napjainkban úgy szokásos megkerülni, hogy a böngészőalapú ügyfelet hoznak létre, esetleg AJAX használatával, és nem ős Windows-ügyfelet alkalmaznak. Pedig az ős Windows-ügyfél AJAX használatával gyakran alkalmasabb válaszképes felhasználói felület létrehozására, mint a böngészők. Az ügyfelek fejlődése okozta kihívások legyőzése érdekében önálló WPF-alkalmazás telepíthető ClickOnce használatával. A ClickOnce-tehcnológia, amely először a .NET 2.0-keretrendszerben jelent meg, lehetővé teszi, hogy az Internet Explorer felhasználói kiválasszanak egy alkalmazást a weben, majd automatikusan telepítsék a helyi gépükre. Az installálás után az alkalmazás képes automatikusan frissülni, amikor új verzió válik hozzáférhetővé. A cél az, hogy kombináljuk egy webügyfél egyszerűségét és olcsó telepítését az önálló WPF-alkalmazás erejével és működésével.

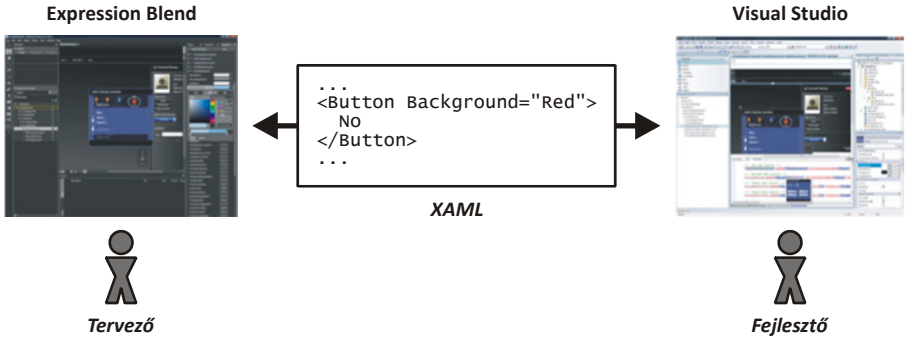
Az önálló WPF-alkalmazások sok esetben jó választásnak bizonyulnak, különösen akkor, amikor ClickOnce használatával telepítjük. A felhasználó betölthet egy XBAP-ot is közvetlenül a böngészőjébe. Ez az alkalmazás majd egy WPF-alapú felhasználói felületet tud nyújtani a felhasználó számára. Minden internetről letöltött XBAP egy részben megbízható tesztágyon fut, ezzel megvédi a felhasználót a rosszindulatú fejlesztőktől. A .NET-keretrendszer által nyújtott kódhozzáféréssel szabályozott adatvédelmen alapulva ez a tesztágy korlátozza, hogy az XBAP mit tehet. A tesztágy által kivetett korlátozások ellenére egy XBAP a WPF-funkciók nagy részét képes használni, beleértve a 2D és a 3D grafikákat, az animációkat, a képernyőn megjelenített dokumentumokat, a képeket, a videókat és a többit.

Ahogy azt korábban leírtuk, a WPF az XMAL FlowDocument elemének segítségével lehetővé teszi az alkalmazásokhoz alkalmazkodó dokumentumok megjelenítését. Azok a dokumentumok, amelyek külső megjelenése azon múlik, hogy hogyan jelenítjük meg őket a képernyőn, nem mindig jelentik a legjobb megoldást. Időnként jobb választást jelenthetnek a fix formátumú dokumentumok, amelyek mindig ugyanúgy néznek ki a képernyőn és nyomtatásban. A WPF XPS dokumentumai megoldják ezt a problémát. Mivel az XAML egy részének használatával hoztuk létre, az XPS-dokumentumok bármely rendszerben olvashatók, amely rendelkezik XPS-olvasóval. Új nyomtatási formátumokat is nyújt a Windows számára, és nagyobb pontossággal teszi lehetővé összetett grafikák nyomtatását.

Eszközök a Windows Presentation Foundation használatához

Lehetőség van bármely WPF felhasználói interfész közvetlen létrehozására kód formájában és/vagy XAML-ben egy alap szövegszerkesztő használatával. A legtöbb ember szeret jó eszközökkel dolgozni, ezért szolgáltatja a Visual Studio 2008 a WPF-tervezőt. Ennek az eszköznek a használatával a fejlesztő grafikusan tudja létrehozni a látni kívánt felhasználói felületet, majd az eszköz generálja a kódot ehhez a felülethez.

A fejlesztők gyakran nem a legmegfelelőbb emberek felhasználói felületek definiálásához. A tervezők gyakran sokkal jobban értenek ehhez, mivel ők az emberekkel való kommunikáció szakértői. A legtöbb tervező nem ír kódot, és így a WPF-tervező nem túl hatékony eszköz számukra. A Microsoft e helyett létrehozta az Expression Blend alkalmazást, amely lehetővé teszi a tervezők számára a hatékony munkát a WPF világában. Az alábbi ábrán látható, hogyan vannak egymással kapcsolatban ezek az eszközök.



1.11. ábra: *A rendelkezésre álló fejlesztőeszközök összefüggése*

Ahogyan az 1.11. ábra is illusztrálja, a tervező az Expression Blendet használhatja annak meghatározására, hogyan nézzen ki és milyen hangulatú legyen a felület animációk specifikálásához, és így tovább, majd az eszköz előállítja ennek egy XAML-verzióját. A fejlesztő importálhatja ezt az XAML-t a Visual Studióba, és hozzáadhat kódot például, eseménykezelés esetére. Mivel mind a Visual Studio, mind az Expression Blend ugyanazt a rendszert használja, lehetőség van arra, hogy a fejlesztő és a tervező felváltva dolgozzon egy projekten, és mindegyik azt az eszközt használja, amiben otthonosan mozog. A cél az, hogy segítsünk az embereknek, hogy a tervezés és a szoftverfejlesztés eltérő szemlélete ellenére hatékonyan dolgozzanak együtt.

A Windows Presentation Foundation és más Microsoft-technológiák

A WPF a többi .NET 3.5-keretrendszer komponensekhez hasonlóan rendelkezik kapcsolattal más Microsoft-technológiákhoz. Ezek közül a legfontosabbak a következők:

- Windows Forms: a .NET-keretrendszer alkalmazások grafikus felületének létrehozására irányuló eredeti szemlélete, a Windows Forms jó néhány használatban lévő alkalmazásban megtalálható. Még a WPF megjelenésével is jó választás maradt a Windows Forms néhány új program létrehozásához, ilyenek például, az üzletviteli (LOB, Line-of-Business) alkalmazások. Annak érdekében, hogy ezek a technológiák együtt használhatóak legyenek, a WPF-alkalmazás hosztol Windows Forms vezérlőelemeket, valamint a Windows Forms alkalmazások képesek hosztolni WPF-vezérlőelemeket. Például, egy Windows Forms alkalmazás hosztolhat egy WPF-vezérlőelemet, amely háromdimenziós adatokat jelenít meg, vagy egy WPF-alkalmazás használhatja a Windows Forms DataGridView vezérlőelemét. Bár van néhány korlátozó tényező, lehetséges olyan alkalmazások létrehozása, amelyek mindkét technológiát használják.
- Silverlight: míg az XBAP lehetővé teszi böngészőben futó WPF-alapú felületek létrehozását az olyan Windows-rendszerek számára, amelyekben a .NET-keretrendszer installálva van, ezek korlátozva vannak. A Rich Internet Application (RIA) tipikusan igényli, hogy hozzáférhető legyen bármilyen típusú ügyfél részéről. Ennek a szükségletnek a kielégítésére kínálja a Microsoft a Silverlightot. Ez a szabadon letölthető technológia a WPF minden funkcióját magában foglalja, beleértve a 2D grafikákat, az animációkat és a videót, és sokféle böngészőben fut Windows-, Macintosh- és (Novell-en keresztül) Linux-rendszer alatt is.
- ASP.NET and ASP.NET AJAX: a Windows-alkalmazások használhatják a WPF-et (vagy esetleg a Windows Formsot) felhasználói felületeikhez, míg a többplatformos RIA a Silverlightot használhatja. Eggyel több lehetőség, hogy szabványos alapokon nyugvó webfelületekről gondoskodjunk csak annak felhasználásával, amit a böngésző nyújt, letöltés nem szükséges. Tulajdonképpen ez az, amit

ASP.NET vagy ASP.NET AJAX segítségével megtehetünk. A felületeknek ez a típusa nem jó arra, hogy gondoskodjon némely RIA-sajátosságról, mint például a videó, de, különösen AJAX használatával, válaszképes és hatásos felhasználói élményt tud nyújtani.

Windows CardSpace

A felhasználók a szokásos módon férhetnek hozzá az alkalmazásokhoz a hálózaton keresztül akár webböngésző, akár más ügyféltípus használatával. Mivel ezek az alkalmazások általában kérik felhasználóiktól, hogy azonosítsák magukat valamilyen módon, a felhasználók, ha távoli alkalmazásokhoz akarnak fordulni, rendszeresen rá vannak szorítva azonosító beszerzésére és bemutatására. Ennek egy igen általános példája az internetalkalmazásokhoz való hozzáférés böngészőn keresztül.

Ahogy korábban már leírtuk, manapság az emberek leggyakrabban felhasználónevekkel és jelszavakkal fejezik ki ezeket a digitális identitásokat, minden problémával együtt, ami ezzel jár. A Windows CardSpace, egy nagyobb azonosító metarendszer része, alternatív szemléletet kínál a probléma megoldására. Hogy jobban megértsük, hogyan is teszi ezt a CardSpace, kezdjük az azonosító metarendszer alapfogalmaival.

A Windows CardSpace és az azonosító metarendszer

Amikor egy felhasználó kapcsolódik egy alkalmazáshoz, akár a webböngészőből, akár egy alkalmazásspecifikus ügyfélről vagy bárhonnan másról, általában megmutatja digitális identitásának valamilyen formáját. Nagyon sokféle digitális identitás fordul elő, de a hálózaton virtuálisan mind egy *biztonsági token (biztonsági jogkivonat)* által jelenik meg. Míg egy egyszerű biztonsági token csak egy felhasználónévből és egy jelszóból áll, addig összetettebb tokenek tartalmazhatnak X.509 tanúsítványt vagy XML-dokumentumot. Bárhogyan is jelenítjük meg őket, a biztonsági tokenek napjaink tipikus mechanizmusai a digitális identitások megjelenítésére a hálózaton.

Míg ez azt próbálja elhitetni velünk, hogy egy napon mindent egy általános biztonsági token formájában fogunk megkapni, a valóság az, hogy különböző szemléletek lesznek továbbra is használatban. Éppen úgy, ahogy manapság többféle azonosító kártyát hordunk magunkkal a tárcánkban – vezetői engedély, hitelkártya stb. –, mindig lesznek a biztonsági tokenek

különböző típusával megjelenített különböző digitális azonosítóink is. Egyetlen azonosító rendszer sem képes mindenre kiterjedő megoldást nyújtani, és így a sokrétű biztonsági tokenek mindig szükségesek lesznek.

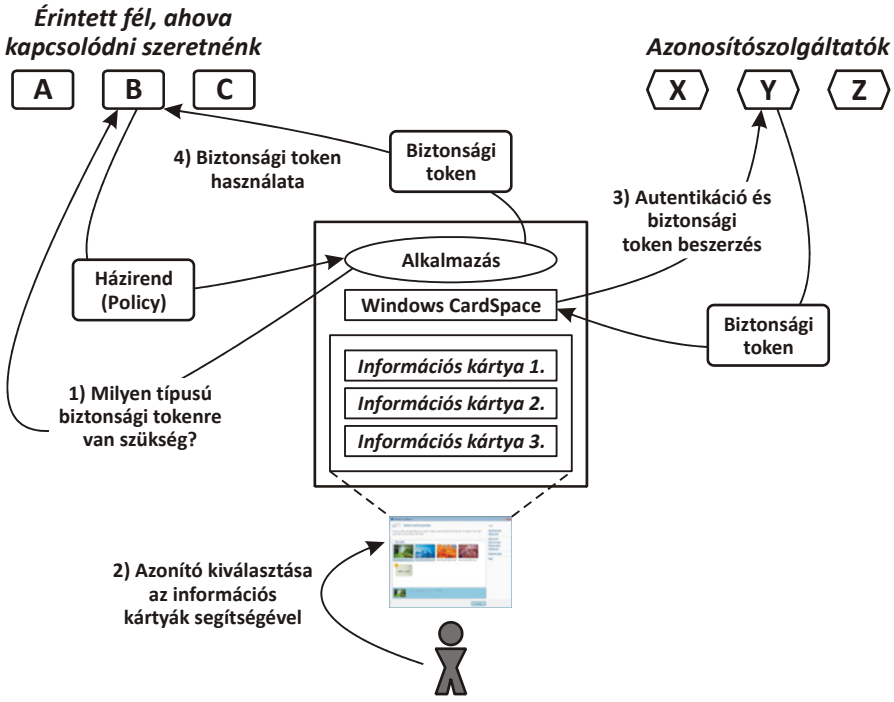
Ezért a felhasználóknak szükségük van valamilyen megoldásra, hogy a különböző digitális azonosítóikkal következetesen tudjanak dolgozni. Bár egyetlen azonosítórendszer nem lesz elég, lehetséges az azonosítórendszerek egy *rendszerének* – egy azonosító metarendszernek – a létrehozása, amely lehetővé teszi digitális azonosítók tízezreinek következetes használatát. Másokkal együttdolgozva a Microsoft vezeti a metarendszer definiálásának a folyamatát. Nyílt webszolgáltatás-technológiákra alapozva, mint például a WS-Security és a WS-Trust, ez a metarendszer meghatározza, hogyan szerezhetjük meg és használhatjuk a digitális azonosítókat, tekintet nélkül a biztonsági token típusára, amelytől függnek.

A digitális identitások kibocsátásának, megszerzésének és használatának folyamata három különböző szerepet igényel. Ezek a szerepek a következők:

- Felhasználó: hivatalosan *tárgy*ként ismertebb, a felhasználó a lényeg, aki rendelkezik digitális identitással.
- Azonosítószolgáltató: ő látja el a felhasználót digitális azonosítóval. Például, a munkához használt digitális azonosító esetében az azonosítószolgáltató a munkaadónk, és a biztonsági tokenünk valószínűleg egy rendszer által keletkezhet, mint például az Active Directory. Az Amazonnal használt digitális azonosító esetében az azonosítószolgáltató ténylegesen mi magunk vagyunk, mivel mi magunk határoztuk meg felhasználónevünket és jelszavunkat. A különböző azonosítószolgáltatók által létrehozott digitális azonosítók különböző információkat hordozhatnak, és különböző szinten tudják biztosítani az általuk tartalmazott információk pontosságát.
- Érintett fél: egy alkalmazás, amely bizonyos módon a digitális identitásra támaszkodik. Az érintett fél gyakran használ azonosítót (azaz az információt ennek az azonosítónak a biztonsági tokenjében tároljuk) egy felhasználó hitelesítésére, majd hoz egy hitelesítési döntést, mint például valamely információ elérésének engedélyezése ennek a felhasználónak. Az érintett fél arra használhatja az azonosítót, hogy megkapjon egy hitelkártyaszámot, hogy ellenőrizze, ugyanaz a felhasználó fért-e hozzá eltérő időben, vagy más célból.

Az érintett fél tipikus példái internetes webhelyek, például bankok, online kereskedések és aukciós oldalak, és bármilyen alkalmazás, amely kéréseket fogad webszolgáltatásokon keresztül.

Ez a három lényeges dolog teremt interakciót az azonosító metarendszerben. Az 1.12. ábra illusztrálja ezeket az interakciókat, valamint azt, hogy hol illeszkedik bele a CardSpace.



1.12. ábra: *Interakció az azonosító metarendszerben*

A folyamat akkor kezdődik, amikor egy Felhasználó CardSpace-t ismerő alkalmazáson keresztül kapcsolódik egy Érintett félhez. Ahhoz, hogy megismerjük, milyen típusú biztonsági tokent igényel ez az Érintett fél, az alkalmazásnak meg kell kapnia házirendjét (1. lépés). Egy webhelyen keresztül elért böngésző esetében, amely napjainkban a legáltalánosabb példa, az oldal házirendje HTML-ben van, és egy weboldal része. Az alkalmazás webszolgáltatáson keresztül elért alkalmazás esetében is inkább WS-MetadatExchange által meghatározott szabvány ipari protokollt használ arra, hogy megkérdezze az érintett féltől a házirendjét. Ebben az

esetben a házirendek egy másik ipari szabvány, a WS-SecurityPolicy segítségével készülnek. Ha a házirendekre vonatkozó információt megszerezzük, ez mindig megmutatja, hogy a biztonsági tokenek milyen fajtáját fogja ez az Érintett fél fogadni, és milyen információkat kell a tokeneknek tartalmazniuk.

Ha már a CardSpace tudja, hogy a biztonsági tokenek milyen fajtájára van szüksége az érintett félnek, megjeleníti a korábban bemutatott azonosítóképernyőt. Minden digitális azonosító, amely a felhasználó számára hozzáférhető, egy információs kártyaként jelenik meg ezen a képernyőn. Külső érintett fél által létrehozott kártyákra *irányított* kártyaként hivatkozhatunk, míg a CardSpace önkibocsátó szolgáltatója által kibocsátottak *önkibocsátott* kártyaként ismertek. A kártyák mindkét típusa megjeleníthető ezen a képernyőn, és a felhasználónak lehetősége van kiválasztani az egyiket a két típus közül. A képernyő, hogy megkönnyítse ezt a döntést, megmutatja, melyik azonosítók felelnek meg az Érintett fél követelményeinek úgy, hogy az alkalmatlanokat kiszürkíti. A Felhasználó ezután kiválasztja ezek közül azt az egyet digitális azonosítóként, amelyet használni szeretne (2. lépés).

A kártya nem tartalmaz aktuális biztonsági tokent. Ehelyett azokat az információkat tárolja, amelyek ahhoz szükségesek, hogy egy bizonyos azonosítószolgáltatót lokalizáljon, és egy biztonsági tokent igényeljen ennek a felhasználónak. (Tulajdonképpen mindegyik kártyát eredetileg valamelyik azonosítószolgáltató hozta létre.) A CardSpace a felhasználó által kiválasztott kártya tartalmát használja arra, hogy biztonsági tokent igényeljen az azonosítószolgáltatótól, amely kibocsátotta a kártyát (3. lépés). Ez a kérés WS-Trust használatával jön létre, amely egy másik szabvány ipari protokoll, és a felhasználó Kerberos, X.509 tanúsítvány és digitális aláírás vagy más mechanizmus használatával hitelesíti magát az azonosítószolgáltató felé. A tokent titkosított formában kapjuk vissza, amely tartalmaz egy időbélyegzőt is, hogy megvédje ezt a tokent attól, hogy ellopják és felhasználják a jövőben.

Ha a kért biztonsági tokent visszakaptuk, elküldjük az Érintett félhez (4. lépés). Az, hogy az érintett fél hogyan használja a tokenben található információt, változtatható. Például, ha a token egy X.509 tanúsítvány tartalmaz, és digitális aláírás kíséri, az Érintett fél valószínűleg a felhasználó hitelesítésére fogja használni, de nem követelmény, hogy a tokent hitelesítésre vagy bármely más biztonsággal kapcsolatos cél érdekében használja. A token olyan információkat hordozhat, mint például egy felhasz-

náló korának bizonyítása, egy internetes bevásárló oldal árengedményének kiválaszthatósága, vagy bármi más. A hitelesítés a biztonsági tokenek egyik fontos használata, de nem az egyetlen lehetőség.

Fontos megjegyezni, hogy a CardSpace-nek nem kell járatosnak lennie a biztonsági tokenekhez használt formátumok és technológiák tekintetében. A metarendszer célja nem az, hogy új egyedi forrásokat próbáljon meg létrehozni digitális identitásokhoz vagy szabványformátumot biztonsági tokenekhez, hanem az, hogy koherens módot nyújtson a biztonsági tokenek bármely fajtáján alapuló digitális identitás használatához. Azzal, hogy a CardSpace a metarendszer fő részének egy Windows-implementációját szolgáltatja, fontos szerepet játszik abban, hogy ezt az általános szemléletet a digitális identitásokhoz valósággá tegye.

Harc az adathalászattal

Az azonosítószolgáltató gyakran különbözik a felhasználótól, mint például amikor egy identitást egy munkaadó engedélyez. De rengeteg olyan helyzet van, ahol az azonosítószolgáltató tulajdonképpen a felhasználó maga. Ha, például, nem használjuk a CardSpace-t valamely webhely eléréséhez, amely kéri felhasználónév és jelszó megadását, akkor mindkettőt a felhasználó definiálja. Ha a felhasználó létrehozta ezt az azonosítót, a felhasználónevet és jelszót később felhasználhatja, amikor ellenőrzi bankköveteléseit, könyvet vesz vagy valami mást csinál, amit az oldal lehetővé tesz.

Ez az egyszerű önkibocsátott azonosító, mivel jelszótól függ, kedvelt célpontja a támadóknak, ahogy azt korábban már leírtuk. A CardSpace a támadások csökkentése érdekében az önkibocsátott azonosító létrehozásának egy alternatív módját nyújtja. Ez az önkibocsátó azonosítószolgáltató helyileg a felhasználó Windows-rendszerében fut. Az önkibocsátó azonosítószolgáltató által létrehozott biztonsági token ahelyett, hogy felhasználónévre és jelszóra támaszkodna, a Security Assertion Markup Language (SAML – biztonsági követelés jelölőnyelv), egy OASIS által meghatározott szabvány használatával van definiálva. Ezek a tokenek jelszavak helyett nyilvánoskulcs-technológiára támaszkodnak a felhasználók azonosítójának érvényesítéséhez, és ha az érintett fél elfogadja azokat, akkor ugyanazt a szerepet játszhatják, mint a hagyományos felhasználónév és jelszó. Az előny az, hogy nincs többé jelszó, amelyet az adathalászok ellophatnak. A jelszavak használatának csökkentése az adathalászat jelentőségét erősen lecsökkentik.

Windows CardSpace és más Microsoft-technológiák

A CardSpace számos más Microsoft-technológiával kapcsolatban van, beleértve a következőket:

- WCF: a CardSpace a WCF-et használja kommunikáláshoz, mivel webszolgáltatási szabványokra támaszkodik, mint a WS-Security és a WS-Trust. Tulajdonképpen WCF-alkalmazás, létrehozója az alkalmazással végeztetheti a CardSpace használatát azáltal, hogy meghatároz egy egyéni kötést.
- Active Directory: bár ez jelenleg nem lehetséges, az Active Directory egyszer majd képes lesz azonosítószolgáltatóként tevékenykedni a metarendszerben. Ha ez lehetővé válik, valószínűleg láthatjuk majd a Windows CardSpace-t a vállalatokon belül éppen úgy, mint az interneten.
- Windows Live ID: a Microsoft Live ID azonosító rendszere manapság képes úgy működni, mint irányított azonosítószolgáltató a Windows CardSpace-hez. Megjegyezzük, hogy a CardSpace nem helyettesíti a Live ID-t (korábban Passportként volt ismert), mivel a kettő eléggé más problémát old meg.

Összefoglalás

A .NET 3.5-keretrendszer a legújabb megtestesülése a legfőbb Windows programozási környezetnek. Elődeire építve és azt kibővítve a célja az, hogy támogassa a korszerű alkalmazások megalkotását. A Microsoft azzal, hogy a különböző technológiáit egy általános alapra építi rá, igyekszik az egészet többé tenni, mint részek összessége, lehetővé téve a fejlesztők számára alkalmazások létrehozását, amelyek a .NET 3.5-keretrendszer különböző részeit használják koherens módon.

2002-es első megjelenése óta a .NET Framework az új Windows-alkalmazások domináns platformja lett. Napjainkban széles körben használják vállalatok, hardverfüggetlen szoftverfejlesztők és a Microsoft maga. A Framework bebizonyította, mennyit ér. A .NET 3.5-keretrendszer a Visual Studio 2008-cal együtt reprezentálja a Microsoft jelenlegi helyzetét az alkalmazásfejlesztés piacán.