

Kapitola 1

Začínáme se Silverlightem

V Silverlightu lze vytvářet nejenom stylová rozhraní, ale také interaktivní aplikace pro různé prohlížeče. Ale co samotné aplikace v Silverlightu? Ne jen vložit video nebo pěkně vypadající sadu tlačítek, ale chodící, hovořící, plně funkční obchodní aplikaci. Samozřejmě že chcete mít také velmi elegantní rozhraní. Dobrá zpráva je, že díky Silverlightu 2 můžete mít všechno najednou.

V této kapitole se budeme věnovat některým kritickým oblastem, s nimiž byste se měli seznámit, abyste mohli vyvíjet aplikace Silverlightu řízené daty, přičemž součástí budou i požadované instalace a nástroje. Než začnete vyvíjet aplikace pomocí Silverlightu, je důležité pochopit hlavní rozdíly mezi verzemi Silverlightu 1 a 2. Těmito rozdíly se budeme v této kapitole zabývat a dozvíte se, proč je každé zdokonalení v Silverlightu důležité pro vývoj aplikací řízených daty.

Jedním s podstatných vylepšení v Silverlightu 2 je možnost psát kód .NET. Vývojáři tak mohou využívat svých stávajících schopností a vytvářet propracované aplikace Silverlightu. Při vývoji těchto aplikací lze používat i funkce jazyka .NET 3.5 v Silverlightu, jako je například LINQ a implicitní typy proměnných. V této kapitole si některá z těchto rozšíření jazyka .NET 3.5 probereme, protože se s nimi budete setkávat v celé knize. Projdeme si také jednotlivé kroky při vytváření jednoduché aplikace Silverlightu řízené daty.

Význam přístupu k datům

Za pestrými uživatelskými rozhraními, přehráváním videí a stylovými šablonami se v Silverlightu nachází propracovaná struktura, jež dobře spolupracuje s různými typy datových zdrojů. Aplikace Silverlightu běží v prohlížeči klientského počítače, kde mezi nimi je Internet a jakákoliv serverová aplikace. Aplikace Silverlightu jsou odpojeny od vzdálených datových zdrojů, takže musí při odesílání a přijímání dat komunikovat se vzdálenými servery prostřednictvím různých typů webových služeb.

Silverlight nabízí několik způsobů pro získání dat ze vzdálených serverů pomocí HTTP a soketů. K nejoblíbenějším a nejužitečnějším metodám patří komunikace s webovou službou založenou na SOAP na vzdáleném serveru (viz kapitoly 5 a 6) a příjem dat ze služby REST (viz kapitoly 7 až 9). V rámci Silverlightu je k dispozici několik nástrojů a metod pro příjem dat, práci s daty, vázání dat, prezentaci uživateli a zpracování všech aspektů aplikace řízené daty.

Většina aplikací vyžaduje určitý typ interakce dat. Data mohou pocházet z databáze, informačního kanálu RSS, webové služby nebo služby REST, jež vrací data ve formátu Plain Old XML (POX). Aplikace Silverlightu mohou různým způsobem komunikovat se službami a vracet data. Ačkoli může být Silverlight velmi obsáhlým a výkonným nástrojem pro konečné uživatele, pro aplikaci jsou důležitá data a jejich doručování.

V posledních letech získával přístup k datům různé významy. Termín *přístup k datům* skutečně popisuje přístupování k datům z libovolného počtu zdrojů. To může znamenat přístup k datům z databáze přímo prostřednictvím ADO.NET, ADO.NET Entity Framework, NHibernate, LBLGen Pro, Enterprise Library nebo nějakého vlastního obchodního objektu a vrstvy přístupu k datům. Může jít také o přístup k datům prostřednictvím webových služeb, ASMX, Windows Communication Foundation (WCF), informačních kanálů RSS, webových služeb ve stylu REST a požadavků HTTP. K datům lze tedy přistupovat různě a po jejich získání je často nutné data určitým způsobem zpracovat. K uspořádání získaných dat do vhodnější formy vyvojáří často využívají LINQ. Všechny tyto faktory definují *přístup k datům* a všechny postupy lze využívat přímo či nepřímo pomocí Silverlightu.

Jdeme na to

Silverlight je skvělý nástroj určený pro návrhy propracovaných rozhraní, jež fungují v různých prohlížečích. Silverlight je díky rozsáhlé podpoře programování možným řešením pro obchodní aplikace. V rámci této knihy se seznámíte s vytvářením vizuálně a funkčně vyzrálých aplikací prostřednictvím Silverlightu.

Než začneme s příklady, je třeba nainstalovat a nakonfigurovat programy potřebné pro Silverlight 2. Nainstalujte si verzi Silverlight 2 runtime, abyste mohli zobrazovat a spouštět aplikace Silverlightu v prohlížeči. Verze runtime funguje (a je oficiálně podporována) v prohlížečích Internet Explorer, Google Chrome, Firefox a Safari. Runtime je k dispozici pro operační systém Windows a Mac OS X. Silverlight 2 podporují následující prohlížeče:

- Microsoft Internet Explorer verze 6.0, 7.0 a 8.0 beta
- Mozilla Firefox verze 1.5, 2.0 a 3.0
- Apple Safari verze 2.0 a 3.0 beta
- Google Chrome

Nástroj Silverlight 2 Tools je k dispozici jako doplněk Visual Studia 2008 a umožňuje vytvářet aplikace Silverlightu 2 pomocí .NET a Silverlightu 2. Při instalaci doplňku Silverlight Tools

se nainstaluje i Silverlight 2 runtime a Silverlight 2 SDK. Silverlight 2 SDK obsahuje příklady, dokumentaci a nástroje pro vývoj aplikací Silverlightu.

Pro vývoj a práci se Silverlightem 2 jsou vyžadovány následující komponenty. Všechny jsou součástí *jednoho* instalačního souboru:

- Silverlight 2 Runtime.
- Silverlight 2 SDK.
- Silverlight 2 Tools for Visual Studio.

Níže uvedené nástroje pro ladění mohou velmi pomoci při hledání potíží v komunikaci aplikací a služeb Silverlightu 2. Všechny jsou bezplatné a mohou vyjasnit důvod potíží a pomohou určit vhodná řešení. V příloze B naleznete tipy a triky pro práci s těmito nástroji při ladění komunikace mezi Silverlightem a službami.

- Firebug (pro Firefox).
- Web Development Helper (pro Internet Explorer).
- Fiddler2 (pro veškeré zachytávání síťového provozu).

Doporučuje se také instalace Visual Studia 2008 a Expression Blend s SP1. Visual Studio 2008 je skvělé pro úpravy XAML, využívání IntelliSense a samozřejmě pro psaní kódu .NET pro aplikace Silverlightu. Expression Blend se skvěle hodí pro uspořádání složitých návrhů a rozvržení v rámci Silverlightu. Jedná se o hodnotné nástroje, jejichž funkcí byste měli při vývoji se Silverlightem využívat.

Funkce Silverlightu 2

Silverlight 2 s sebou přináší podporu pro kód .NET v aplikacích Silverlightu, nejde však o jediné podstatné zdokonalení. V tabulce 1.1 je uveden přehled hlavních vylepšení v Silverlightu 2, přičemž některá z nich jsou základem pro vývoj aplikací řízených daty pomocí Silverlightu. Tyto funkce jsou v tabulce 1.1 zvýrazněny, protože je třeba seznámit se s tím, jak mohou vývojářům pomoci při vytváření spolehlivých aplikací řízených daty se Silverlightem.

Tabulka 1.1. Klíčové funkce Silverlightu

Nové funkce v Silverlightu	Poznámky
Podpora jazyků a technologie .NET Framework	Silverlight podporuje kompatibilní část .NET Framework a programovací jazyky Visual Basic, Visual C#, IronRuby a IronPython.
Datové vazby a oznamování	V Silverlightu jsou nyní k dispozici datové vazby a oznamování založené na XAML s využitím rozhraní <code>INotifyCollectionChanged</code> a <code>INotifyPropertyChanged</code> .

Tabulka 1.1. Klíčové funkce Silverlightu

Nové funkce v Silverlightu	Poznámky
Izolované úložiště	Silverlight 2 může ukládat informace v chráněné oblasti na klientském počítači.
Smluvní a mimosmluvní datové služby: webové služby RSS založené na JSON, REST, SOAP, POX a Atom	Prostřednictvím Silverlightu jsou k dispozici různé datové služby. To zjednodušuje čtení dat z webových služeb.
Síťový přístup do různých domén	Silverlight 2 může přistupovat ke službám, jež nepocházejí ze serveru s aplikací Silverlightu2.
LINQ, lambda výrazy, rozšiřující metody a inicializátory objektů	Součástí Silverlightu je LINQ to Objects, LINQ to XML a LINQ to JSON sloužící k definování dotazů pro složité datové struktury.
Podpora rozložení (layout) StackPanel, Grid a Canvas	Silverlight 2 podporuje tři významné panely rozložení nativní pro WPF a XAML.
Sada ovládacích prvků a panelů s možností vazby	Silverlight 2 přináší celou řadu ovládacích prvků, jež lze integrovat do aplikací Silverlightu 2 a lze vytvářet jejich vazby s datovými zdroji.



Společnost Microsoft chtěla původně vydat Silverlight 2 jako Silverlight 1.1, avšak po všech provedených zdokonaleních společnost usoudila, že nová verze je dostatečně odlišná na to, aby si zasloužila své vlastní číslo verze.

Když se podíváte na funkce uvedené v tabulce 1.1, je zřejmé, že Silverlight 2 přináší mnoho významných nových schopností. Je důležité, abyste se seznámili s každým vylepšením, jež přímo ovlivňuje vývoj aplikace řízené daty. Mnoho z nich si v následujících kapitolách podrobně popíšeme, nyní si je tedy pouze stručně představíme.

Podpora jazyků a technologie .NET Framework

Vývoj aplikace Silverlightu je jednoznačně mnohem jednodušší, pokud můžete využít své zkušenosti s vývojem .NET. Počínaje schopností psát kód .NET v klientské aplikaci Silverlightu pro zpracování různých aspektů dané aplikace, včetně obsluhy událostí, přes využívání rozsáhlé platformy .NET Framework a vytváření složitých uživatelských ovládacích prvků. Silverlight 2 runtime obsahuje malou, ale výkonnou podmnožinu knihovny .NET Framework, takže vývojáři mohou využívat své stávající znalosti .NET a přejít na Silverlight s vynaložením minimálního úsilí.

Model pro popis webových služeb a jejich operací

Silverlight může komunikovat s webovými službami založenými na SOAP (např. prostřednictvím WCF nebo ASMX) a předávat definované datové struktury mezi klientem Silverlightu a vzdáleným serverem. Tyto služby vystaví s využitím jazyka WSDL kontrakt, s nímž může klient komunikovat vytvářením tříd proxy. Tento kontrakt definuje služby, jež lze volat, a způsob jejich volání. Vystavuje také datové struktury, jež lze předávat do služeb a ze služeb. To je klíčové pro vystavování webových služeb, jež vrací nebo přijímají serializovatelné jednotky jako parametry. Díky smluvnímu rozhraní může Silverlight komunikovat se vzdálenými službami, jež vracejí entity z LINQ to SQL, NHibernate, ADO.NET Entity Framework, nebo i vlastní modely doménových entit. Umožňuje také komunikaci se službami třetích stran založenými na SOAP, jež vystavují WSDL, jako je například Live Search.

Volná vázanost datových služeb

Silverlight může využívat datové služby také bez vytváření tříd proxy prostřednictvím SOAP, RSS, Atom, JSON, POX nebo REST. K dispozici je mnoho zdrojů informací, jež vrací data prostřednictvím nesmluvní služby. Například Digg.com (<http://www.digg.com>), Amazon.com (<http://www.amazon.com>), Twitter.com (<http://www.twitter.com>) a Flickr.com (<http://www.flickr.com>) vystavují služby REST, jež vrací XML. Aplikace Silverlightu mohou požadovat informace ze všech těchto služeb a výsledky pak integrovat do aplikace. Těmto oblastem se budeme velmi podrobně věnovat v kapitolách 7 až 9, přičemž se také dozvíte, jak s nimi lze komunikovat mezi různými doménami a jak zpracovat XML a JSON, jež mohou vrátit prostřednictvím LINQ to XML a LINQ to JSON.

Nový model ovládacích prvků

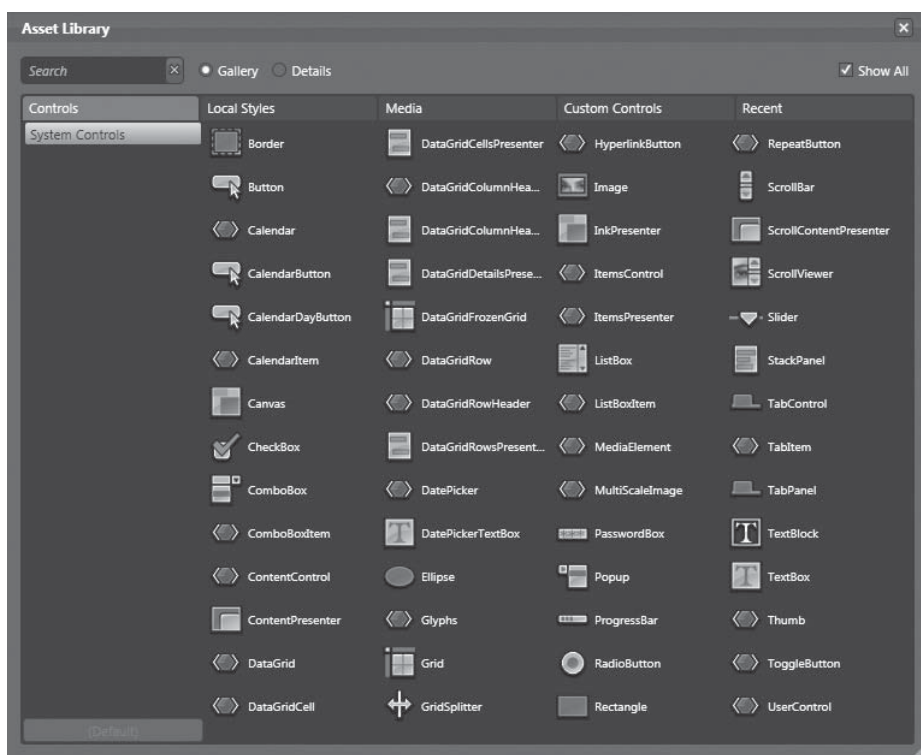
Nový model ovládacích prvků v Silverlightu 2 velmi usnadňuje vytváření aplikací Silverlightu. Nové ovládací prvky lze uspořádat v rámci panelů rozvržení (layout) XAML, jako je `Grid`, `Canvas` a `StackPanel` a vytvářet tak propracovaná rozhraní. Ovládací prvky podporují vlastnosti datových vazeb a závislostí založených na XAML, jimž se budeme podrobně věnovat v kapitolách 2 až 4.

Silverlight 2 přináší více než dvacet ovládacích prvků, s jejichž pomocí lze vytvářet klientské aplikace Silverlightu. Nabízí panely rozvržení (layout), jako je například `Grid`, `Canvas` a `StackPanel` a několik ovládacích prvků pro vstup, jež lze upravovat s využitím šablon a stylů. Obsah některých ovládacích prvků lze zcela nahradit a vytvořit tak složitější řešení. Můžete například nahradit záhlaví ovládacího prvku `Grid` řadou ovládacích prvků, jež umožní seřadit `DataGrid`. Lze také nahradit výchozí vzhled tlačítka jiným prvkem, jako je `Ellipse` a vyplnit jej dalším prvkem `FrameworkElement`, například obrázkem. S ovládacími prvky se blíže seznámíte v kapitolách 2 až 4, v nichž je také vysvětleno, jak s nimi fungují datové vazby a jaké jsou nejvhodnější postupy pro vazby a prezentace dat. Na obrázku 1.1 jsou znázorněny ovládací prvky dostupné

v Expression Blend 2 se SP1 a na obrázku 1 je přehled ovládacích prvků, jež jsou k dispozici ve Visual Studiu 2008.

LINQ to Objects a LINQ to XML

Silverlight 2 podporuje využívání LINQ to Objects, což může zjednodušit dotazování dat v polích a kolekcích v aplikaci Silverlightu. LINQ to XML (také podporovaný v aplikacích Silverlightu 2) je velmi užitečný při využívání nesmluvních datových služeb, jež vrací XML. XML lze načítat do různých knihoven, avšak využití objektů LINQ to XML umožňuje dotazovat XML prostřednictvím známé a výkonné syntaxe LINQ. Pomocí LINQ lze také kombinovat XML a objektové datové struktury ve stejném dotazu LINQ. V mnoha příkladech uvedených v této knize budeme LINQ v určité podobě používat k dotazování dat z XML nebo objektů.



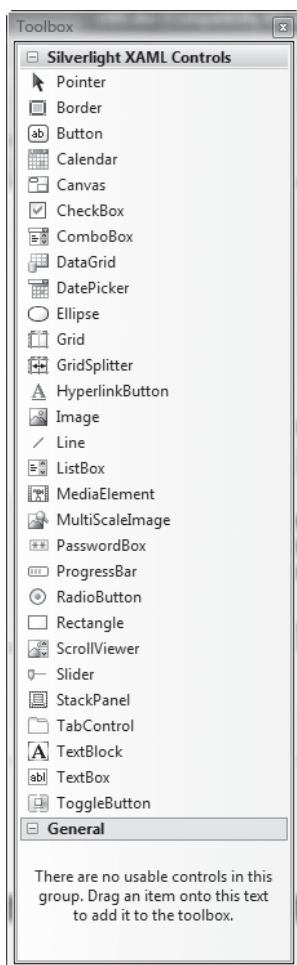
Obrázek 1.1 Ovládací prvky Silverlightu 2 zobrazené v Expression Blend

LINQ

Aplikace řízené daty vytvořené v Silverlightu ve velké míře využívají rozšíření jazyka LINQ a .NET 3.5, jak si ukážeme v mnoha příkladech v této knize. Tyto funkce jsou důležité a stojí

tedy za to seznámit se s tím, jak fungují. V jazycích C# 3 a Visual Basic (VB) 9 byla představena rozšíření, z nichž některá jsou základní pro psaní dotazů s LINQ. Aplikace Silverlightu mohou využívat například XML, JSON, objekty, entity z LINQ to SQL a entity z ADO.NET Entity Framework a další.

Aplikace Silverlightu často potřebují získávat hodnoty z polí a (nebo) seznamů vlastních entit (např. `List<T>`). V těchto případech je LINQ to Objects užitečný, protože umožňuje dotazovat jakýkoli seznam `IEnumerable`. Ačkoli lze XML využívat a spravovat prostřednictvím různých knihoven .NET, jako je například `XmlReader`, LINQ to XML nabízí jednodušší a často výkonnější způsob využití XML.



Obrázek 1.2 Ovládací prvky Silverlightu 2 zobrazené ve Visual Studiu 2008

Aplikacím Silverlightu to usnadňuje využívání XML ze zdrojů RSS nebo služeb REST a zpracovávání dat, aniž by došlo k rozvlácným smyčkám. Dotazy LINQ se mohou také připojit k se-

znamu objektů a XML, což je užitečné v případě, že aplikace získává data z více zdrojů. V několika kapitolách této knihy budeme pracovat s LINQ to Objects a LINQ to XML na příslušných místech, kde jich mohou aplikace Silverlightu využívat.

LINQ to Entities se velmi často používá při dotazování modelu Entity Data Model, který je vytvářen ADO.NET Entity Framework. ADO.NET Entity Framework vytváří Entity Data Model, jenž mapuje objektově orientovaný pojmový model entit do relačního úložiště, jako je například SQL Server (ačkoli ADO.NET Entity Framework podporuje model poskytovatelů, takže jsou podporovány i další databázové servery, jako je Oracle). Dotazy se píšou pomocí LINQ to Entities proti pojmovému modelu a dotazy se překládají prostřednictvím mapovací vrstvy na příkazy SQL, jež se provádějí oproti relačnímu úložišti. Po vrácení entity z dotazu LINQ to Entities lze entity serializovat a předat do dalších vrstev aplikace. Klientská aplikace Silverlightu by například mohla požadovat entitu ze služby WCF na vzdáleném serveru, jenž vystavuje entity z modelu Entity Data Model. Aplikace Silverlightu by pak na tyto entity mohla použít také LINQ to Objects.

Čtvrtou podobou LINQ, které se budeme v této knize věnovat, je LINQ to SQL. LINQ to SQL umožňuje mapování 1:1 z entit na objekty SQL Server. Nepodporuje tak obsáhlé schéma mapování jako ADO.NET Entity Framework; nemá ani model poskytovatelů, takže funguje pouze s SQL Serverem. Pomocí LINQ to SQL můžete psát například dotazy vracející entity, jež lze serializovat a předat do aplikace Silverlightu.

Rozšíření jazyka

Verze .NET 3.5 s sebou přinesla několik rozšíření jazyka .NET. Tato rozšíření jsou spolu s funkcemi důležitou součástí tvorby dotazů pomocí LINQ.

V této části si ukážeme několik klíčových funkcí jazyka, včetně následujících:

- Automatické settery (metoda pro zápis)/getter (metoda pro čtení) vlastností.
- Inicializátory objektů.
- Inicializátory kolekcí.
- Metody rozšíření.
- Implicitní typy proměnných.
- Anonymní typy.

Rozšíření jazyka .NET 3.5 jsou znázorněna na obrázku 1.3.

Automatické vlastnosti v C#

Vytváření vlastností může být velmi redundantní proces, zvláště pokud getter a setter neobsahují jinou logiku, než je získávání a nastavování hodnoty privátního pole. Použití veřejných polí by sice zkrátilo délku kódu, tato pole však mají několik nevýhod. Nejpodstatnější nevýhodou je to, že veřejná pole nejsou plně podporována vazbami dat.

Jednou z možností, jak se vyhnout psaní kódu pro privátní pole a jeho veřejný getter a setter vlastností, je použití refaktorizačního nástroje. Automatické možnosti však umožňují psát méně kódu a přesto získat privátní pole a jeho veřejný getter a setter.

```
List<Customer> customerList = new List<Customer>
(
    new Customer { ID = 101, CompanyName = "Foo Company",
                  BusinessAddress = new Address { State="FL" } },
    new Customer { ID = 102, CompanyName = "Goo Company",
                  BusinessAddress = new Address { State="NY" } },
    new Customer { ID = 103, CompanyName = "Hoo Company",
                  BusinessAddress = new Address { State="NY" } },
    new Customer { ID = 104, CompanyName = "Koo Company",
                  BusinessAddress = new Address { State="NY" } }
);

var query = from c in customerList
            where c.BusinessAddress.State.Equals("FL")
            select new { Name = c.CompanyName,
                        c.BusinessAddress.State };

```

← Inicializátor kolekce

← Inicializátor objektu

← Vložený inicializátor objektu

← Anonymní typ

Implicitně typovaná proměnná

Obrázek 1.3 Přehled rozšíření jazyka .NET 3.5



C# automatické vlastnosti podporuje, VB nikoli.

Pomocí zkrácené syntaxe vytvoříte automatickou vlastnost. Kompilátor pak za vás vygeneruje privátní pole a veřejný setter a getter. V příkladu 1.1 mají třídy Customer2 a Address2 několik privátních polí, jež jsou vystavena prostřednictvím řady odpovídajících veřejných getterů a setterů vlastností.

Příklad 1.1 Explicitní gettersy a settersy v C#

```
public class Customer2
{
    private int id;
    private string companyName;
    private Address2 companyAddress;

    public int ID
    {
        get { return id; }
        set { id = value; }
    }
}
```

```

    }
    public string CompanyName
    {
        get { return companyName; }
        set { companyName = value; }
    }

    public Address2 CompanyAddress
    {
        get { return companyAddress; }
        set { companyAddress = value; }
    }
}
public class Address2
{
    private int id;
    private string ci ty;
    private string state;
    public int ID
    {
        get { return id; }
        set { id = value; }
    }

    public string Ci ty
    {
        get { return ci ty; }
        set { ci ty = value; }
    }

    public string State
    {
        get { return state; }
        set { state = value; }
    }
}

```

Příklad 1.2 ukazuje, jak lze stejného výsledku dosáhnout pomocí automatických vlastností, přičemž je třeba napsat méně kódu, než v příkladu 1.1. Třídy `Customer` a `Address` používají automatické vlastnosti k vytvoření vlastností tříd a nevyžadují tolik kódu pro definování pole a jeho getteru a setteru vlastností.

Příklad 1.2 Automatické vlastnosti

```
public class Customer
```

```

{
    public int ID { get; set; }
    public string CompanyName { get; set; }
    public Address CompanyAddress { get; set; }
}

public class Address
{
    public int ID { get; set; }
    public string City { get; set; }
    public string State { get; set; }
}

```

Inicializátory objektů

Inicializátory objektů umožňují předávat pojmenované hodnoty pro každou veřejnou vlastnost, jež pak bude použita k inicializaci objektu. Pomocí kódu uvedeného v příkladu 1.3 byste například mohli inicializovat instanci třídy `Customer`.

Příklad 1.3 Vytváření a inicializace třídy

```

C# Customer customer = new Customer();
customer.ID = 1001;
customer.CompanyName = "Foo";
customer.CompanyAddress = new Address();

```

```

VB Dim customer As New Customer()
customer.ID = 1001
customer.CompanyName = "Foo"
customer.CompanyAddress = New Address()

```

Pokud však využijete inicializátory objektů, můžete vytvořit instanci třídy `Customer` pomocí syntaxe uvedené v příkladu 1.4.

Příklad 1.4 Využití inicializátorů objektů

```

C# Customer customer = new Customer
{
    ID = 1001,
    CompanyName = "Foo",
    CompanyAddress = new Address()
};

```

```

VB Dim customer = New Customer() With _

```