edice PROVODCE

Microsoft

Visual Studio .NET

praktické programování krok za krokem

František Šíma, David Vilímek

- Vytvoření nové aplikace
- Návrh databáze
- DataGrid a validace
- Propojení tabulek
- Příprava počítače
- Slovník pojmů

GRADA

Upozornění pro čtenáře a uživatele této knihy

GRADA®

Obsah

Úvod

Začínáte-li pracovat s nějakým produktem, musíte předem vědět, zda je vhodný pro cíl, kterého chcete dosáhnout. **Co** všechno od něj můžete očekávat.

Vyhovuje-li nabízený produkt vašim představám, začne vás zajímat způsob jeho použití. Potřebujete znát, **jak** s tímto nástrojem zacházet, aby se váš záměr povedl. Během času budete zlepšovat svoje dovednosti v dané oblasti, a pak mnohé z vás napadne seznámit se s používaným produktem hlouběji. Budete hledat odpovědi na otázky začínající slovem **proč**.

Tento úvod, spolu s kapitolou 4. Návrh databáze, by vám měl sdělit, **co** produkt umožňuje. Zbytek publikace je zaměřen hlavně na otázku, **jak** toho dosáhnout. Zájemci v něm najdou i odpovědi na svá **proč**.

Zaměřili jsme se na vytváření databázových aplikací v prostředí tenkého klienta. Tato část totiž bývá v různých knihách zmiňována pouze okrajově a programátor, který se z nejrůznějších důvodů rozhodne vytvářet aplikace v tomto prostředí, je nucen naučit se i klienta tlustého.

Aplikace vytvářené pomocí programu Visual Studio jsou typu klient-server. Server je počítač, na kterém je uložena databáze a který představuje datovou vrstvu. Jeho jediným úkolem je efektivní přístup k datům uloženým v databázi. Klient je počítač, který obsahuje vlastní aplikaci (předává příkazy pro server) a který, v případě tlustého klienta, představuje aplikační vrstvu.

Tenký klient je další, třetí vrstvou. Došlo totiž k rozdělení aplikační vrstvy na funkční a prezentační. Jedná se o vizuální část s minimální funkcionalitou omezenou na rychlé ošetření vstupů a na vizualizaci dat. Vlastní funkcionalita je na aplikačním serveru a její část může být až na databázovém serveru. Jak je vidět, pojem tenkého klienta je úzce spjat s trojúrovňovou architekturou. V našem konkrétním případě se jedná o webový prohlížeč na straně tenkého klienta.

Při práci s tenkým klientem jste omezeni možnostmi webového prohlížeče, tj. možnostmi HTML. Ten neumožňuje využití všech ovládacích prvků Windows. Jaký smysl má vlastně použití tenkého klienta, když jeho možnosti z pohledu uživatele jsou menší? Hlavní výhoda spočívá v údržbě a v instalaci u uživatelů. U klientských počítačů stačí, když mají nainstalován webový prohlížeč. Instalovat aplikaci tedy znamená, že uživatel obdrží pouze webovou adresu stránky, ze které se má aplikace vyvolat, a ta je tím nainstalována. Údržbu aplikace pak stačí provádět pouze na aplikačním serveru. Nároky na počítače u uživatelů mohou být též menší.

V této publikaci budeme vytvářet typickou databázovou aplikaci, která by vás v průběhu její tvorby měla seznámit se všemi podstatnými částmi týkajícími se Visual Studia při hromadném zpracování dat. Po jejím přečtení byste měli být schopni vytvářet databázové aplikace, pracující s několika vzájemně provázanými tabulkami.

I když se budeme snažit své programy neustále zdokonalovat podle toho, jak porostou naše znalosti, neznamená to, že následující program bude vždy zahrnovat všechny dosud získané poznatky. Na stránce **DPH.aspx** se například seznámíme se způsobem, jak ověřovat data dříve, než se je pokusíme zapsat do databáze. V následujících programech se k nim ale už nevrátíme. Chceme totiž aby program zůstal přehledný a aby obsahoval pouze probíranou funkčnost. Při psaní skutečné aplikace byste ovšem měli data ověřovat vždy.

Simulovali jsme situaci, kdy na začátku je pouze vhodný počítač a na konci je aplikace „Fakturace“, kterou jste schopni distribuovat. Aby tvrzení, obsažená v této knize, odpovídala realitě, nainstalovali jsme na prázdný počítač program Visual Studio .NET a začali jsme vytvářet ukázkovou aplikaci, která by vás měla průběžně seznamovat se všemi podstatnými částmi tohoto produktu, jež jsou nutné při vyvíjení robustních databázových aplikací. Současně vznikala tato publikace. Zaznamenávali jsme do ní všechny situace, se kterými jsme se v průběhu vývoje naší úlohy střetli, i když se netýkaly přímo Visual Studia .NET. V případě, že bylo třeba rozhodnout se mezi efektivitou pro uživatele a jednoduchostí z hlediska programátora, volili jsme jednoduchost. Protože jednoduchost je spojena s vizuálním programováním, dáváme přednost i vizuálnímu programování. Toto slovo je ostatně obsaženo i v názvu produktu. Efektivní programy mají tu nevýhodu, že jsou náročné na údržbu.

Jednotlivé příkazy, prvky apod. jsme uváděli tak, aby nebyly použity samoúčelně, ale v kontextu s vytvářenou aplikací.

Vzorová databáze „Fakturace“ obsahuje pouze 5 tabulek (z nichž jedna představuje tabulku parametrů), na kterých jsme se snažili ukázat základní vazby, se kterými byste měli uspět i při vytváření vazeb složitějších.

Ukázková aplikace provádí evidenci došlých faktur s vazbou na číselník organizací. U faktur evidujeme i řádky, které mají vazbu na číselník přijatých zdanitelných plnění. Faktury jsou

číslovány vzestupnou řadou s tím, že poslední číslo faktury udržujeme v parametrickém souboru. Mezi fakturami a číselníkem organizací je použita neidentifikační vazba, mezi fakturami a řádky vazba identifikační. Tato aplikace by mohla sloužit jako evidence DPH nebo jako vstup do účetnictví.

V první části publikace se ještě nepředpokládá, že jste seznámeni s logikou práce ve Visual Studiu. Proto jsme zvolili strategii „přilepování" funkčnosti. Výsledný program vznikne po několika průchodech programováním. Při prvním průchodu se spokojíme s pouhým překladem zatím nefunkčního programu, v dalším pak se zobrazením dat a po dalším rozvíjení programu se dostaneme ke konečnému kroku s hotovou funkční aplikací pro aktualizaci. Vzhledem k tomu, že jednu a tutéž věc lze mnohdy řešit několika způsoby, snažíme se např. aktualizaci provádět pokaždé jinak. Při tvorbě profesionální aplikace by měly mít všechny programy stejnou formu.

V další části, kdy již předpokládáme, že základy práce ovládáte, budeme postupovat přesně obráceně. Ukážeme si výsledný program a potom rozebereme jeho jednotlivé části.

Publikace je určena pro všechny programátory, kteří již pracovali s nějakým jazykem (včetně programování v DOS) nebo alespoň tuší co to je podmínka, cyklus či sekvence, kteří mají představu co je SQL a kteří zběžně ovládají Windows. Po přečtení byste měli být schopni napsat robustní databázovou aplikaci s využitím transakcí.

Vzhledem k tomu, že tato kniha má formu učebnice, čtěte ji souvisle tak jak je napsána. V každé kapitole se totiž seznámíte s určitými pojmy, znalostmi a postupy, na něž pak budeme navazovat. Vysvětlíme-li si například postup jak „Nakonfigurovat SqlDataAdapter", nebudeme tento postup znova vysvětlovat, pouze uvedeme „Nakonfigurujte SqlDataAdapter". Na konci každé kapitoly se budeme snažit vás na nové pojmy, znalosti a postupy upozornit.

Doporučujeme též, abyste si po každé části zkusili samostatně vytvořit svůj vlastní příklad. Naučíme-li se například jak vytvořit tabulku, zkuste si podle uvedeného postupu zhotovit svojí vlastní tabulku.

1.

Nová aplikace (projekt)

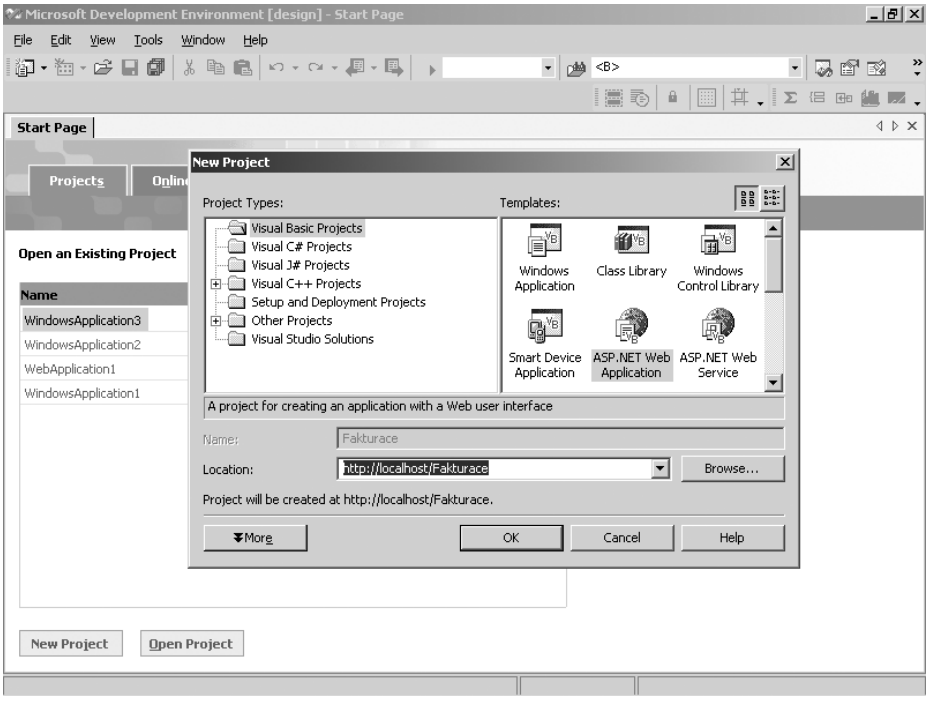
V této kapitole se naučíme, jak vytvořit novou aplikaci, přeložit ji a vzápětí vyvolat. Vzhledem k tomu, že tato publikace je zaměřena na tenkého klienta, bude naše aplikace prezentována webovou stránkou. Její snadné rutinní volání umožníme prostřednictvím programu Microsoft Internet Explorer tím, že ji zahrneme mezi oblíbené položky. I když aplikace v této kapitole nebude mít zatím žádnou funkčnost, uvedený postup využijeme i při vytváření složitějších aplikací. V prostředí Visual Studia se aplikace nazývá projekt, a proto budeme používat i toto označení. Pojem aplikace budeme používat v uživatelském smyslu, pojem projekt ve smyslu vývojářském.

Projekty ve Visual Studiu mohou být různých typů. V následujících kapitolách jsme vybrali jako typ projektu Visual Basic Projects, neboť se nám zdál nejsrozumitelnější pro všechny ty, kteří mají zkušenost s nějakým jiným jazykem. Programátoři C si pravděpodobně vyberou typ projektu Visual C#.

Po vyvolání Visual Studia se zobrazí posledně použité 4 projekty (jestliže existují). Programátor má možnost vyvolat již existující projekt nebo vytvořit projekt nový. Protože právě začínáme a budeme vytvářet nový projekt, stiskneme tlačítko **New Project**. Dále, po stisknutí tlačítka **OK**, se zobrazí dialogové okno pro výběr již zmíněného typu

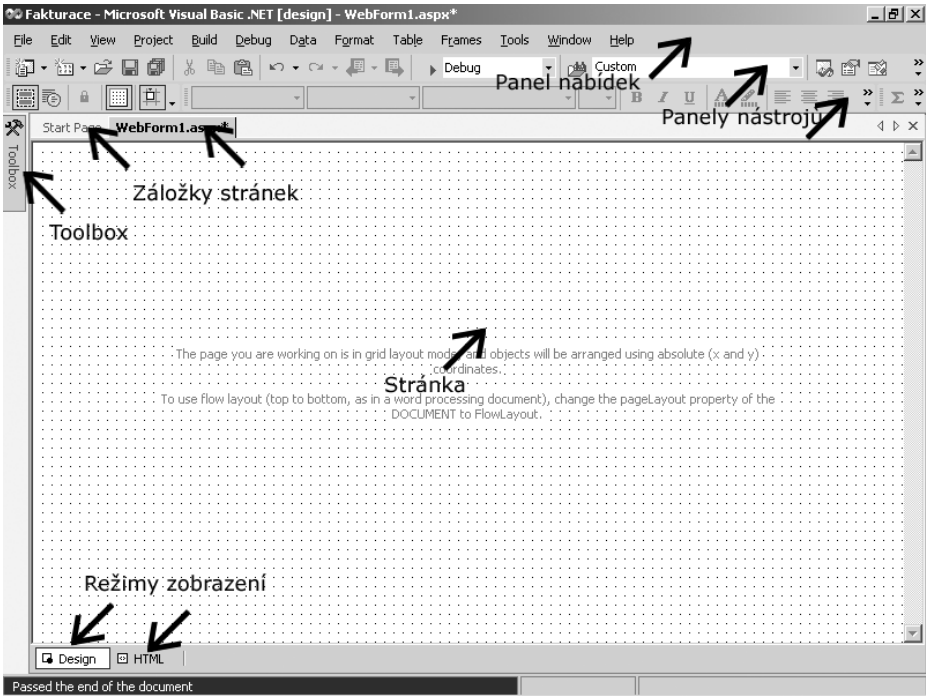
projektu (viz obrázek 1.1). V závislosti na typu projektu lze vybrat i typ zobrazení (šablonu), kterým se budou prezentovat data na obrazovce – **Template**. Protože vytváříme projekt ve Visual Basicu, vybereme **Visual Basic Projects** a pro již zmíněného tenkého klienta vybereme následně aplikaci **ASP.NET Web Application**. V rozvíracím seznamu je zobrazen navrhovaný název projektu spolu s umístěním. Obsluha má možnost tento název i umístění změnit. Rozhodli jsme se změnit pouze název. Protože vytváříme aplikaci pro evidenci přijatých faktur, přepsali jsme navrhovaný název *http://localhost/WebApplication1* na *http://localhost/fakturace*.

*Označení projekt je v tomto případě poněkud nepřesné. Nejedná se totiž o projekt, ale o řešení (solution). Obecně může totiž řešení obsahovat více projektů, i když v případě této publikace bude obsahovat vždy pouze jediný. Toto „nepřesné" označení jsme použili proto, že je použito na úvodní obrazovce ve Visual Studiu. Jak ukazuje obrázek 1.1, používají se termíny **New Project**, **Open Project**, **Open an Existing Project**, ale ve skutečnosti se jedná o řešení.*

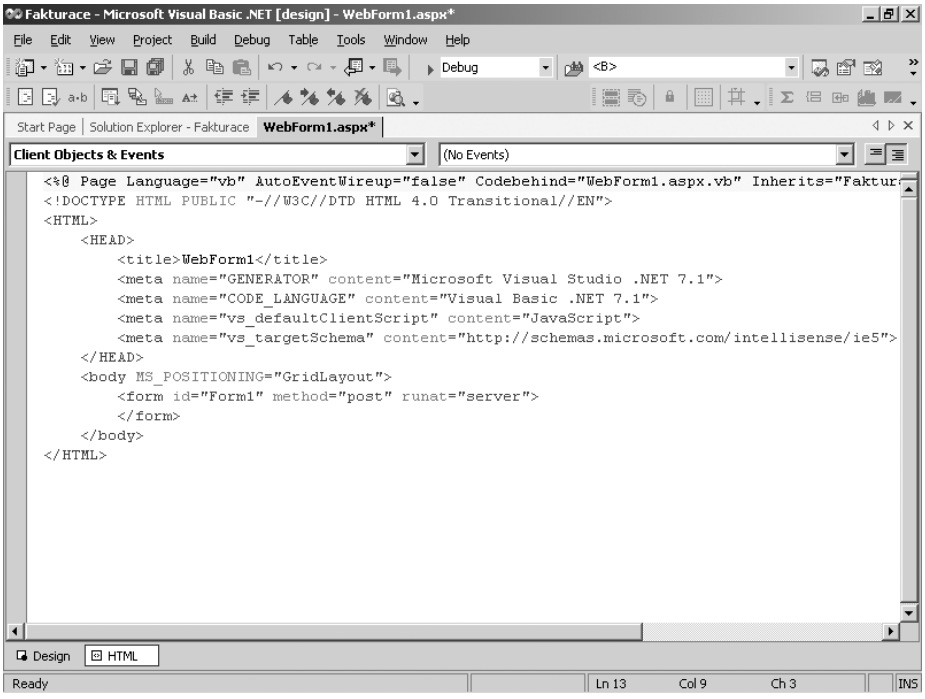


Obr. 1.1: *Výběr typu projektu*

Po stisknutí tlačítka **OK** se zobrazí pracovní plocha pro návrh první webové stránky (viz obrázek 1.2). V horní části (tak jako u všech produktů Microsoft) jsou pod **panelem nabídek** (**MenuBar**) umístěny **panely nástrojů** (**Toolbars**). Pod panely nástrojů jsou umístěny záložky jednotlivých stránek (v našem případě zatím pouze již zmiňovaná pracovní plocha navrhované webové stránky s názvem **WebForm1.aspx** a úvodní stránka Visual Studia **Start Page**). Po levé straně je minimalizované okno **Toolbox** a ve spodní části je záložka s dotazem na způsob zobrazení pracovní plochy stránky. Pracovní plocha každé navrhované webové stránky může být totiž zobrazena ve dvou režimech (modech), a to buď v režimu **Design**, nebo v režimu **HTML**. Změny provedené na jedné záložce se



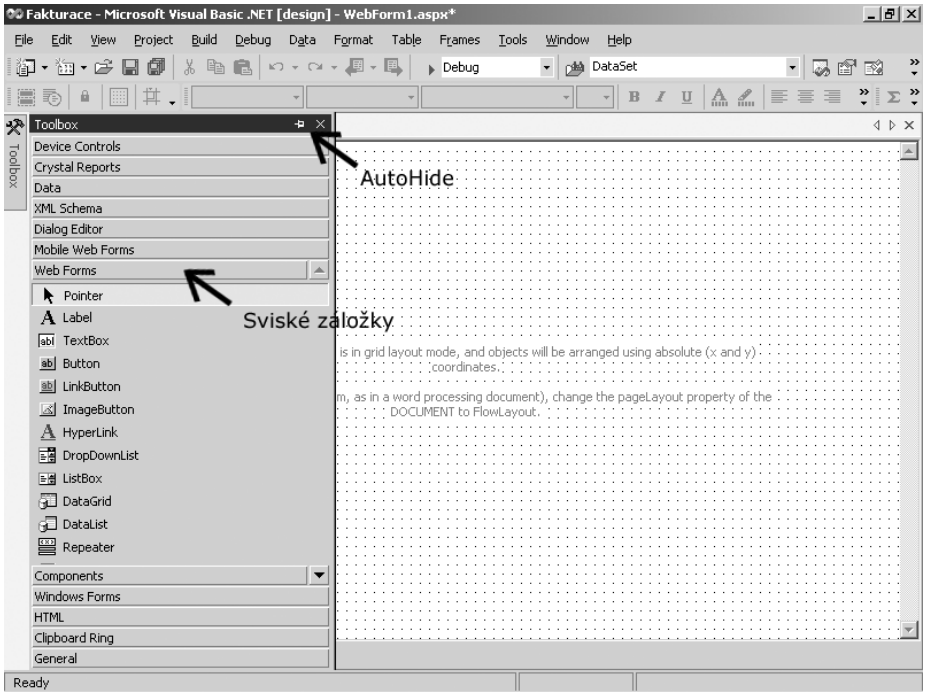
Obr. 1.2: *Pracovní plocha pro návrh webové stránky v režimu Design*



Obr. 1.3: *Pracovní plocha pro návrh webové stránky v režimu HTML*

okamžitě promítají do druhé záložky a vy si můžete vybrat, ve kterém režimu si přejete pracovat. My budeme pracovat v návrhovém režimu (**Design**), který zobrazuje pracovní plochu v takové podobě, jaká bude po vyvolání stránky uživatelem, nicméně pro ilustraci si zobrazíme i režim **HTML** (viz obrázek 1.3). Měl-li by vás zobrazený režim **HTML** odradit od dalšího čtení, nemusíte se ničeho bát. Můžete se bez něj obejít. Zvláště nyní, když s Visual Studiem teprve začínáme.

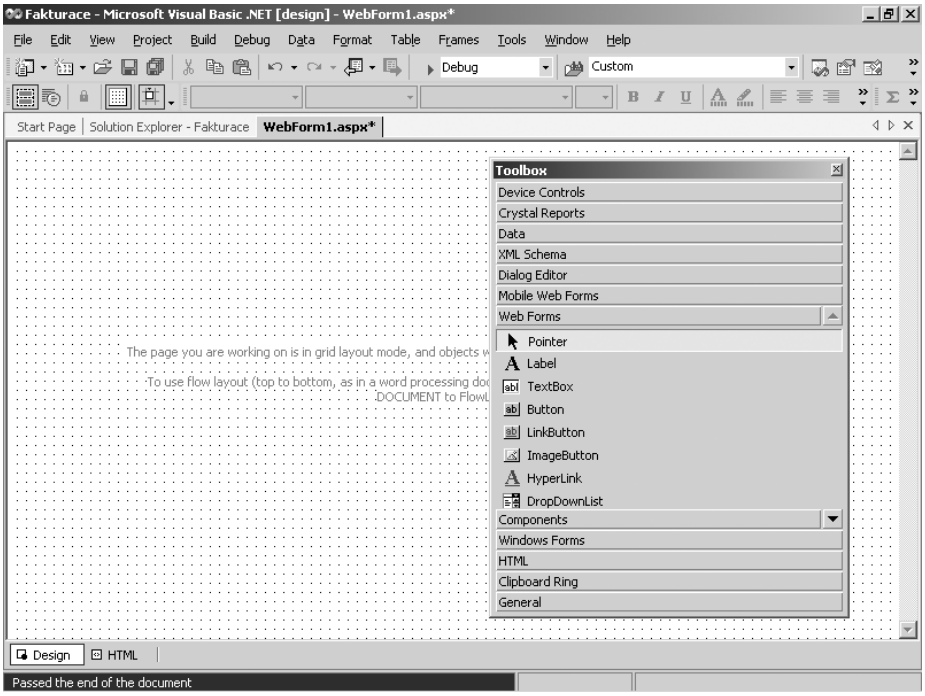
Ukážeme-li myší nad již zmíněný **Toolbox**, „vyjede“ tento do strany, zvětší se a zobrazí se nám jeho nabídka (viz obrázek 1.4). Pomocí okna **Toolbox** můžeme umístit na pracovní plochu různé prvky, jako jsou například pole pro vstup údajů **TextBox**, políčka pro zaškrtnutí **CheckBox** sloužící pro zadávání voleb, seznamy **ListBox**, přepínače **RadioButton** pro alternativní zadávání voleb apod. Komu nevyhovuje tento způsob práce s oknem **Toolbox**, může okno **Toolbox** přetáhnout na pracovní plochu, na které bude otevřené stále. Pomocí myši lze upravit jeho velikost a umístění.



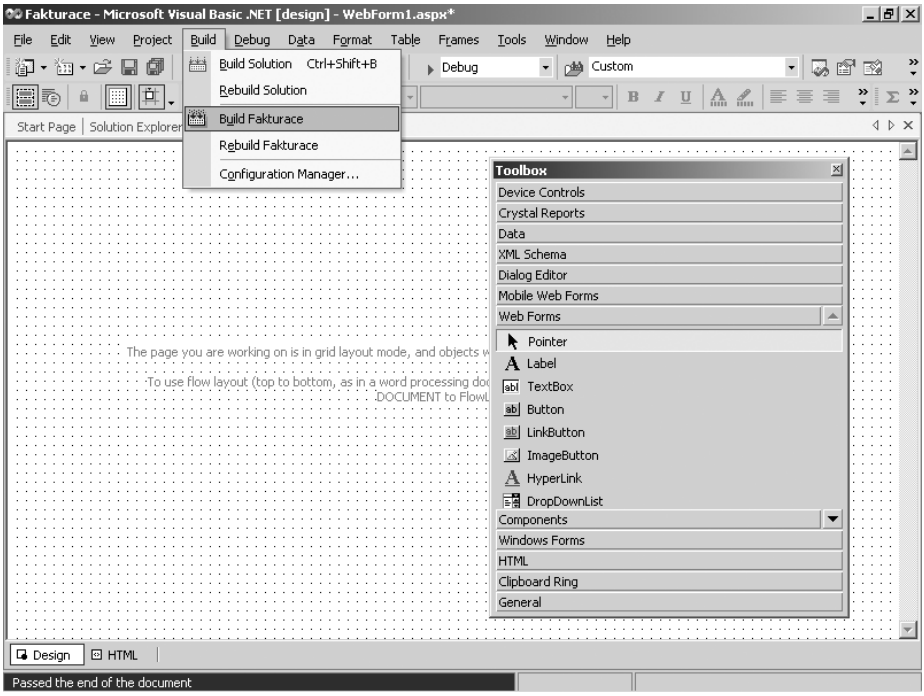
Obr. 1.4: *Okno Toolbox na pracovní ploše v režimu AutoHide*

Okno **Toolbox** přetáhnete na plochu tak, že po jeho „vyjetí“ stisknete tlačítko **AutoHide**, které se nachází vedle křížku v záhlaví okna **Toolbox**, a pak jej uchopením přetáhnete na zvolené místo (viz obrázek 1.5). Tím se dostane do režimu **Float**. V záhlaví tohoto okna můžete nastavit i další vlastnosti. Podaří-li se vám okno **Toolbox** zavřít nebo pokud se vám na pracovní ploše neobjeví, vyvolejte ho zadáním příkazu **View → Toolbox**.

Projekt přeložíme a spustíme, i když zatím ještě nic neumí. Překlad provedeme tak, že klepneme na příkaz **Build** (pátá volba) a potom na libovolnou 1. až 4. položku z nabídky. My jsme jej vyvolali příkazem **Build → Build Fakturace** (viz obrázek 1.6).



Obr. 1.5: *Okno Toolbox na pracovní ploše v režimu Float*



Obr. 1.6: *Překlad projektu*