

# KAPITOLA 17

## Navigace po webu

Navigace je fundamentální komponentou jakéhokoliv webu. Přestože je dost snadné přenést uživatele z jedné stránky na jinou, vytvořit unifikovaný navigační systém, který bude dobře fungovat po celém vašem webu, už zabere trochu práce. Ačkoliv vlastní navigační systém s několika odkazy si můžete vyrobit i sami, nepochybně brzy zjistíte, že to půjde mnohem rychleji, když využijete zabudovaný navigační systém ASP.NET.

V této kapitole se budete zabývat třemi klíčovými tématy:

- **Ovládací prvky MultiView a Wizard.** Umožňují zhustit několik kroků do jediné stránky. S nimi se dá zkombinovat několik pracovních stránek na jediné místo, čímž se navigace zjednodušuje.
- **Plán, neboli mapa webu (site map).** Umožňuje definovat navigační strukturu webu a přímo ji svázat s bohatě vybavenými ovládacími prvky. Dozvíte se také, jak se dá tento pracovní rámec rozšířit, aby podporoval různé typy ovládacích prvků a různá umístění úložišť pro plán webu.
- **Bohatě vybavené navigační ovládací prvky.** Patří mezi ně TreeView a Menu. Přestože navigace není jediným účelem těchto ovládacích prvků, hodí se pro potřeby navigace přímo ideálně. V této kapitole se dozvíte o široké škále jejich funkcí.

Pomocí výše zmíněných ovládacích prvků, plánu webu a vzorů stránek dokážete vybudovat kompletní navigační systém s minimálním úsilím. A nejlepší na tom je, že ASP.NET jasně odděluje data (informace o struktuře webu) od jejich implementace (navigační ovládací prvky). To znamená, že můžete změnit uspořádání webových stránek, nahrazovat je jinými nebo je přejmenovávat, aniž byste web nějak poškodili nebo museli upravovat nějaký kód. Nebudete muset udělat nic víc, než provést potřebné změny v souboru plánu webu vaší aplikace.

### Stránky s více zobrazeními

V mnohých webech se složitější úkoly člení do několika stránek. Pokud například chcete do internetového obchodu přidat položku do nákupního vozíku a dojet k pokladně, musíte přeskočit z jedné stránky na jinou. Je to nejčistší přístup, který se také snadno programuje – za předpokladu, že přenášíte informace z jedné stránky na jinou pomocí vhodné techniky správy stavu (dotazovacími řetězci počínaje a stavem relace konče).

V jiných situacích chcete vložit do jediné stránky kód pro několik jiných stránek. Například byste rádi poskytl výběr jedné z několika variant vztahujících se k zobrazení stejných dat (jako třeba tabulku založenou

na mřížce, nebo graf) a umožnili uživateli se přepínat z jednoho zobrazení do jiného, aniž by musel opustit aktuální stránku. Nebo byste chtěli zpracovat úlohu složenou z několika kroků (klasickým příkladem jsou informace získávané od uživatele při jeho registraci), aniž byste se museli starat o to, jak přenášet relevantní informace mezi jednotlivými webovými stránkami.

---

**TIP** *Z hlediska uživatele asi není velký rozdíl v tom, zdali použijete několik samostatných stránek nebo jedinou stránku s několika zobrazeními. Ve webu, který je dobře navržen, uživatel zaznamená jediný rozdíl – při přístupu přes více zobrazení se nezmění URL adresa v prohlížeči. Primární rozdíl je v modelu kódu. Pokud použijete přístup přes více stránek, získáte dokonalejší separaci, ovšem budete mít více práce s tím, jak mají jednotlivé stránky spolu komunikovat (tedy určit způsob, jakým budou stránky sdílet nebo přenášet informace). U přístupu přes několik zobrazení na jediné stránce sice o tuto separaci přijdete, ale snadněji se vám bude psát kód, který je složen z malých, nedělitelných úloh.*

---

V ASP.NET 1.x se dala stránka s více zobrazeními vytvořit pouze tak, že se na ni přidalo několik ovládacích prvků `Panel`. Každý panel reprezentoval jedno zobrazení (resp. jeden krok). Pak jste mohli nastavovat vlastnost `Visible` jednotlivých panelů, aby v daném okamžiku byl viditelný jenom jeden z nich. Nevýhodou tohoto přístupu je, že stránka se trochu komplikuje, protože se musí psát kód navíc pro správu jednotlivých panelů. A kromě toho – není to zrovna moc robustní přístup, protože drobná chyba může způsobit, že na stránce se zobrazí dva panely současně.

Dnes už naštěstí nemusíte navrhovat vlastní systém několika zobrazení úplně od počátku. Můžete využít jeden ze dvou ovládacích prvků vyšší úrovně, které návrh tohoto druhu značně usnadňují – jsou to prvky `MultiView` a `Wizard`.

## Ovládací prvek `MultiView`

`MultiView` je ze dvou ovládacích prvků pro systém více zobrazení tím jednodušším. `MultiView` v podstatě dává možnost deklarovat několik zobrazení a v daném okamžiku zobrazit pouze jediné. Nemá žádné výchozí uživatelské rozhraní – získáte pouze HTML kód a ovládací prvky, které mu dodáte. `MultiView` je ekvivalentní přístupu přes vlastní panely, který jsme vysvětlili výše.

`MultiView` se vytvoří docela snadno. Do vašeho souboru `.aspx` přidejte značku `<asp:MultiView>`, a pak jednu značku `<asp:View>` pro každé samostatné zobrazení.

```
<asp:MultiView ID="MultiView1" runat="server">
  <asp:View ID="View1" runat="server">...</asp:View>
  <asp:View ID="View2" runat="server">...</asp:View>
  <asp:View ID="View3" runat="server">...</asp:View>
</asp:MultiView>
```

Dovnitř značek `<asp:View>` následně vložte HTML kód a webové ovládací prvky, které budou tvořit dané zobrazení.

```
<asp:MultiView ID="MultiView1" runat="server" ActiveViewIndex="0">
  <asp:View ID="View1" runat="server">
    <b>Showing View #1<br />
    <br />
    <asp:Image ID="Image1" runat="server" ImageUrl="~/cookies.jpg" /></b>
  </asp:View>
```

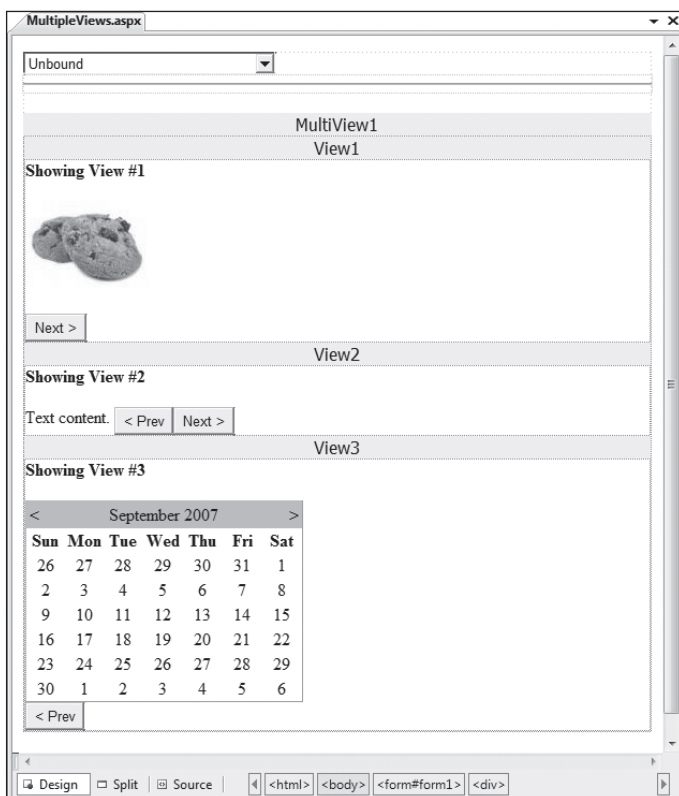
```

<asp:View ID="View2" runat="server">
    <b>Showing View #2</b><br />
    <br />
    Text content.
</asp:View>
<asp:View ID="View3" runat="server">
    <b>Showing View #3</b><br />
    <br />
    <asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
</asp:View>
</asp:MultiView>

```

**TIP** Zobrazení se rovněž dají přidávat programátorsky (stejně jako jakýkoliv jiný ovládací prvek) tím, že vytvoříte instanci nového objektu zobrazení a přidáte ji do `MultiView` metodou `Add()` nebo `AddAt()` kolekce `Views`.

Visual Studio ukáže všechna zobrazení už v době návrhu – pěkně jedno po druhém (viz obrázek 17-1). Jednotlivé regiony můžete editovat úplně stejně, jako byste editovali jakoukoliv jinou část stránky.



Obrázek 17-1. Několik zobrazení v době návrhu.

**POZNÁMKA** *Podobného efektu jako s MultiView docílíte i s ovládacím prvkem Accordion (harmonika), který je součástí ASP.NET AJAX Control Toolkit. Ovládací prvek Accordion umožňuje vytvořit skupinu panelů, které lze sbalovat a rozbalovat. Funguje to tak, že uživatel klikne na záhlaví nějakého panelu, který se následně rozbalí, přičemž všechny ostatní panely se pak automaticky zavřou (sbalí). Accordion má ovšem dramaticky odlišné vnitřnosti než prvek MultiView, protože většinu své práce vykonává na straně klienta. O tomto prvku se dozvíte více v kapitole 32.*

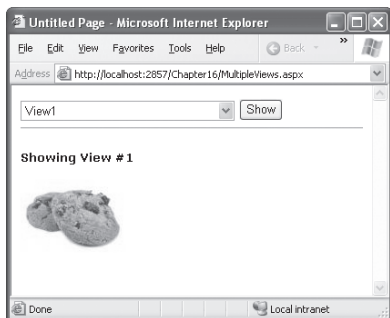
To, co se konkrétně zobrazí, určuje MultiView.ActiveViewIndex. Je to jediné zobrazení, které je ve stránce realizováno. Výchozí hodnota ActiveIndex je -1, což znamená, že se žádné zobrazení neukáže. Je možné to udělat třeba tak, že necháte uživatele, aby si vybral zobrazení z nějakého seznamu, který obsahuje všechna dostupná zobrazení. Podívejte se na úsek kódu, který sváže všechna zobrazení s ovládacím prvkem seznamu:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        DropDownList1.DataSource = MultiView1.Views;
        DropDownList1.DataTextField = "ID";
        DropDownList1.DataBind();
    }
}
```

A tady máte kód, který nastaví aktuální zobrazení podle toho, co uživatel vybere (na základě indexu vybrané položky seznamu):

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    MultiView1.ActiveViewIndex = DropDownList1.SelectedIndex;
}
```

Výsledek vidíte na obrázku 17-2.



**Obrázek 17-2.** Přepínání zobrazení pomocí ovládacího prvku pro seznam.

Pokud chcete dát zobrazením nějaké výstižnější názvy, naplňte jednoduše celý seznam ručně. Jenom dejte pozor, aby souhlasilo pořadí názvů s pořadím zobrazení.

Uvedený kód se vlastně vůbec nemusí psát, protože MultiView obsahuje několik inteligentních značek. Podobně jako některé bohatě vybavené datové ovládací prvky, rozpoznává i MultiView konkrétní názvy pří-

kazů v ovládacích prvcích tlačítek. (Ovládacím prvkem tlačítka se rozumí jakýkoliv ovládací prvek, který implementuje rozhraní `IButtonControl`, takže mezi ně patří `Button`, `ImageButton` a `LinkButton`.) Přidáte-li do zobrazení takový ovládací prvek tlačítka, který bude používat dohodnuté názvy příkazů, bude mít tlačítko jistou automatickou funkcionalitu. Všechny dohodnuté názvy příkazů jsou vypsané v tabulce 17-1. Každý název příkazu má také odpovídající statický člen ve třídě `MultiView`, takže se velmi snadno dostanete ke správnému příkazu, pokud ho budete chtít nastavit programátorsky.

**Tabulka 17-1. Dohodnuté názvy příkazů pro `MultiView`.**

Název příkazu	Člen ve třídě <code>MultiView</code>	Popis
<code>PrevView</code>	<code>PrevViewCommandName</code>	Přejde na předchozí zobrazení.
<code>NextView</code>	<code>NextViewCommandName</code>	Přejde na následující zobrazení.
<code>SwitchViewByID</code>	<code>SwitchViewByIDCommandName</code>	Přejde na zobrazení s konkrétním ID (řetězec s názvem). ID se vezme z vlastnosti <code>CommandArgument</code> ovládacího prvku tlačítka.
<code>SwitchViewByIndex</code>	<code>SwitchViewByIndexCommandName</code>	Přejde na zobrazení s konkrétním číselným indexem. Index se vezme z vlastnosti <code>CommandArgument</code> ovládacího prvku tlačítka.

Pokud si to chcete vyzkoušet, přidejte do prvních dvou zobrazení následující tlačítko (nezapomeňte na to, aby tlačítka měla různý identifikátor, ID):

```
<asp:Button ID="cmdNext" runat="server" Text="Next" CommandName="NextView" />
```

Pak přidejte do druhého a třetího zobrazení toto tlačítko:

```
<asp:Button ID="cmdPrev" runat="server" Text="Prev" CommandName="PrevView" />
```

A nakonec zajistěte, aby se v rozvracím seznamu zobrazil správný název zobrazení, když budete pracovat s tlačítky. Tohle zařídíte v obsluze události `MultiView.ActiveViewIndexChanged`:

```
protected void MultiView1_ActiveViewChanged(object sender, EventArgs e)
{
    DropDownList1.SelectedIndex = MultiView1.ActiveViewIndex;
}
```

### VÝKON STRÁNEK MULTIVIEW

Nejdůležitější věcí, kterou musíte vědět o `MultiView`, je to, že na rozdíl od bohatě vybavených datových ovládacích prvků (`GridView`, `FormView` atd.), `MultiView` není kontejner, jenž by pojmenovával prvky, které v sobě obsahuje. To znamená, že přidáte-li do jednoho zobrazení ovládací prvek s názvem `textBox1`, nemůžete už do jiného zobrazení přidat jiný ovládací prvek se stejným názvem. A skutečně – v termínech modelu stránky není žádný opravdový rozdíl mezi ovládacími prvky, které přidáte do zobrazení, a ovládacími prvky ve zbytku stránky. V obou případech budou ovládací prvky, které vytvoříte, přístupné přes členské proměnné ve vaší třídě stránky.

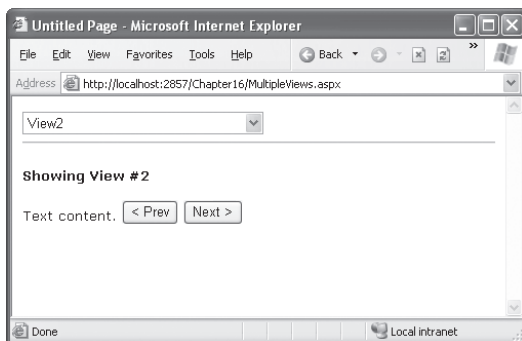
... pokračování z předchozí stránky.

To znamená, že ve druhém zobrazení je možné snadno nakonfigurovat ovládací prvek poté, co byla v prvním zobrazení vyvolána nějaká událost nějakého ovládacího prvku.

Důsledkem je, že stránky, které vytváříte s `MultiView`, mají tendenci se chovat jako normální stránky. Je tomu tak proto, že celý model ovládacích prvků (včetně všech ovládacích prvků ze všech zobrazení) se vytváří při každém odeslání stránky na server a přetrvává ve stavu zobrazení. Většinou to nebude představovat nějaký významný faktor, pokud programátorsky nemanipulujete s velkým počtem ovládacích prvků (v takovém případě by asi bylo vhodné vypnout u těchto ovládacích prvků `EnableViewState`), nebo pokud nepoužíváte několik zdrojů dat. Pokud například máte tři zobrazení a každé z nich má jiný ovládací prvek zdroje dat, tak pokaždé, když se stránka odešle na server, provedou své dotazy všechny tři ovládací prvky zdroje dat, což znamená, že všechna zobrazení se sváží (včetně těch, co nejsou na stránce viditelná). Abyste se vyvarovali této zbytečné zátěže, můžete využít techniky, které jsme popsali v kapitole 9 – například můžete ponechat ovládací prvky nesvázané, takže v případě potřeby je svážete programátorsky, nebo zrušíte proces vázání u těch zobrazení, jež nejsou aktuálně viditelná.

Samozřejmě – nikde není řečeno, že každý `MultiView` musí používat vázání dat. Perfektním scénářem pro `MultiView` je nějaký dlouhý sled vstupních ovládacích prvků – například nějaký formulář, který je použit pro průzkum veřejného mínění, jenž je rozdělen do několika zobrazení, aby se uživatel nemusel po formuláři posouvat nahoru a dolů. Zde funguje `MultiView` uspokojivě, protože nakonec, až vyplňování skončí, snadno přečtete všechna data ze všech ovládacích prvků ze všech zobrazení.

Nyní můžete přecházet z jednoho zobrazení do jiného pomocí tlačítek (viz obrázek 17-3).



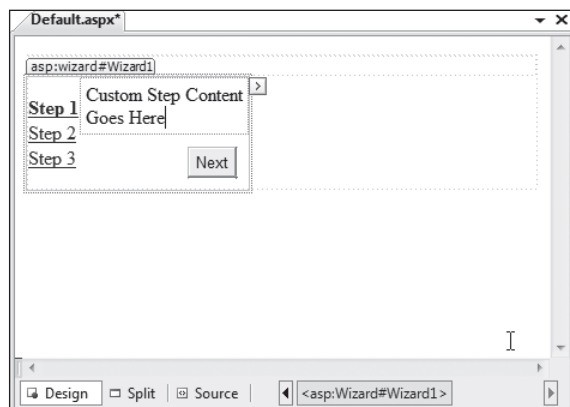
Obrázek 17-3. Přepínání zobrazení pomocí dohodnutých názvů příkazů.

## Ovládací prvek Wizard

Ovládací prvek `Wizard` (průvodce) je atraktivnější varianta ovládacího prvku `MultiView`. Kromě toho, že rovněž podporuje zobrazení vždy jediného panelu z několika, obsahuje i velké množství zabudovaného (a přizpůsobitelného) chování, včetně navigačních tlačítek, stylů, šablon, či pruhu s odkazy na kroky průvodce.

Průvodci obvykle reprezentují jedinou úlohu, ve které uživatel postupně přechází z jednoho kroku na druhý: například z aktuálního kroku přejde na krok bezprostředně následující (nebo na krok bezprostředně předcházející, pokud chce opravit nějaký údaj). Ovládací prvek `Wizard` ASP.NET dále podporuje nelineární navigaci, čímž se zde myslí to, že se můžete rozhodnout ignorovat krok, který je založen na informacích dodaných koncovým uživatelem.

Ovládací prvek Wizard standardně zobrazuje navigační tlačítka a svislý pruh vlevo s odkazy na jednotlivé kroky. Pruh s těmito odkazy můžete skrýt, nastavíte-li vlastnost `Wizard.DisplaySideBar` na `false`. Obvykle se to dělá tehdy, když chcete striktně vynutit, aby uživatel postupoval průvodcem postupně (neskákal v něm sem a tam, jak se mu zachce). Obsah jednotlivých kroků dodáváte prostřednictvím libovolných ovládacích prvků HTML nebo ASP.NET. Na obrázku 17-4 vidíte ukázkou regionu, v němž můžete do instance průvodce, který obsahuje nějaký předem připravený obsah, přidávat váš vlastní obsah.



Obrázek 17-4. Region pro vložení obsahu konkrétního kroku.

## Kroky průvodce

Chcete-li v ASP.NET vytvořit průvodce, jednoduše specifikujte jednotlivé kroky a jejich obsah značkami `<asp:WizardStep>`. Každý krok přebírá nějaké základní informace, které jsou vypsány v tabulce 17-2.

Tabulka 17-2. Vlastnosti třídy `WizardStep`.

Vlastnost	Popis
Title	Popisný název kroku. Tento název se používá ve svislém pruhu odkazů vlevo.
StepType	Typ kroku jako hodnota výčtu <code>WizardStepType</code> . Hodnota určuje typ navigačních tlačítek, která se v daném kroku zobrazí. Možnosti jsou <code>Start</code> (zobrazí tlačítko <code>Next</code> ), <code>Step</code> (zobrazí tlačítka <code>Next</code> a <code>Previous</code> ), <code>Finish</code> (zobrazí tlačítka <code>Finish</code> a <code>Previous</code> ), <code>Complete</code> (nezobrazí žádné tlačítko a skryje pruh odkazů, pokud byl zapnutý) a <code>Auto</code> (typ kroku se odvodí z pozice v kolekci). Výchozí je <code>Auto</code> , což znamená, že první krok je <code>Start</code> , poslední krok je <code>Finish</code> , a všechny ostatní kroky mezi nimi jsou <code>Step</code> .
AllowReturn	Vyjadřuje, zdali se bude uživatel moci vrátit do daného kroku. Jinak řečeno – pokud je <code>false</code> , uživatel se nebude moci vrátit k předchozímu kroku. Odkaz v levém sloupci pro předchozí krok nebude mít pro tento krok žádný efekt a tlačítko <code>Previous</code> v následujícím kroku buď takový nevratný krok přeskočí, nebo bude rovnou skryté (což závisí na hodnotě vlastnosti <code>AllowReturn</code> předchozích kroků).

Následující ukázkou průvodce obsahuje čtyři kroky, které dohromady reprezentují jednoduchý dotazník. Na konci je krok s vlastností `StepType` nastavenou na hodnotu `Complete`, ve kterém se vypíše závěrečná zpráva. Navigační tlačítka a pruh odkazů vlevo se přidávají automaticky.

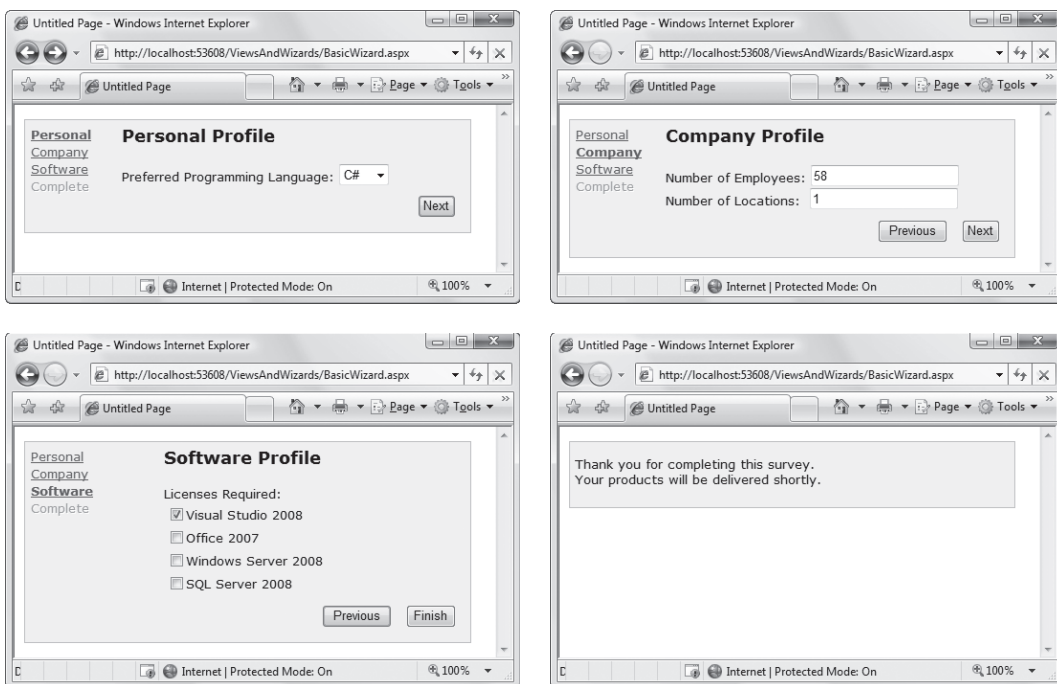
```

<asp:Wizard ID="Wizard1" runat="server" Width="467px"
    BackColor="#EFF3FB" BorderColor="#B5C7DE" BorderWidth="1px">
    <WizardSteps>
        <asp:WizardStep ID="WizardStep1" runat="server" Title="Personal">
            <h3>Personal Profile</h3>
            Preferred Programming Language:
            <asp:DropDownList ID="lstLanguage" runat="server">
                <asp:ListItem>C#</asp:ListItem>
                <asp:ListItem>VB</asp:ListItem>
                <asp:ListItem>J#</asp:ListItem>
                <asp:ListItem>Java</asp:ListItem>
                <asp:ListItem>C++</asp:ListItem>
                <asp:ListItem>C</asp:ListItem>
            </asp:DropDownList>
            <br />
        </asp:WizardStep>
        <asp:WizardStep ID="WizardStep2" runat="server" Title="Company">
            <h3>Company Profile</h3>
            Number of Employees: <asp:TextBox ID="txtEmpCount" runat="server"/>
            Number of Locations: <asp:TextBox ID="txtLocCount" runat="server"/>
        </asp:WizardStep>
        <asp:WizardStep ID="WizardStep3" runat="server" Title="Software">
            <h3>Software Profile</h3>
            Licenses Required:
            <asp:CheckBoxList ID="lstTools" runat="server">
                <asp:ListItem>Visual Studio 2008</asp:ListItem>
                <asp:ListItem>Office 2007</asp:ListItem>
                <asp:ListItem>Windows Server 2008</asp:ListItem>
                <asp:ListItem>SQL Server 2008</asp:ListItem>
            </asp:CheckBoxList>
        </asp:WizardStep>
        <asp:WizardStep ID="Complete" runat="server" Title="Complete"
            StepType="Complete">
            <br />
            Thank you for completing this survey.<br />
            Your products will be delivered shortly.<br />
        </asp:WizardStep>
    </WizardSteps>
</asp:Wizard>

```

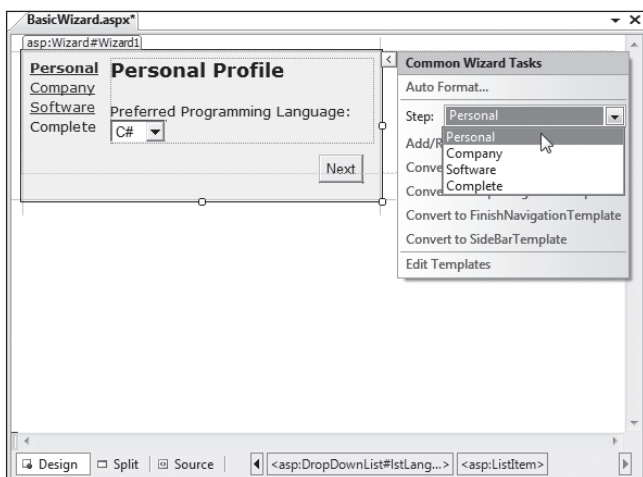
Jednotlivé kroky průvodce vidíte na obrázku 17-5.





Obrázek 17-5. Průvodce skládající se ze čtyř kroků.

Na rozdíl od ovládacího prvku `MultiView` vidíte ve Visual Studiu na návrhové ploše webové stránky vždy pouze jediný krok. Krok, s jehož návrhem chcete právě pracovat, vybíráte z inteligentní značky, viz obrázek 17-6. Ale pozor – pokaždé, když to uděláte, Visual Studio změní hodnotu vlastnosti `Wizard.ActiveStepIndex` na krok, který jste právě vybrali. Před spuštěním aplikace tedy nezapomeňte nastavit hodnotu této vlastnosti zpět na 0, aby průvodce odstartoval prvním krokem.



Obrázek 17-6. Navrhování kroku průvodce.

**POZNÁMKA** Zapamatujte si, že pokud přidáváte ovládací prvky do oddělených kroků průvodce, že u všech se vytvoří instance a že tyto prvky se uchovávají ve stavu zobrazení (view state), bez ohledu na to, který krok je právě aktuální. Potřebujete-li zeštíhlit nějakého složitého průvodce, rozdělte ho do několika samostatných stránek, ze stránky na stránku přecházejte metodou `Server.Transfer()`, a smířte se s méně elegantním programovacím modelem.

## Události průvodce

Pokud to budete potřebovat, můžete vytvořit kód, kterým průvodce přimějete, aby reagoval na několik událostí, jež jsou podrobně vypsány v tabulce 17-3.

**Tabulka 17-3. Události ovládacího prvku Wizard.**

Událost	Popis
<code>ActiveStepChanged</code>	Nastane, když se ovládací prvek přepne do nového kroku (buď proto, že uživatel klikl na navigační tlačítko, nebo pokud jste v kódu programátorsky změnili vlastnost <code>ActiveStepIndex</code> ).
<code>CancelButtonClick</code>	Nastane, když se klikne na tlačítko <code>Cancel</code> . Toto tlačítko se standardně nezobrazuje, nicméně je můžete přidat do každého kroku, nastavíte-li vlastnost <code>Wizard.DisplayCancelButton</code> . V průvodcích obvykle tlačítko s funkcí <code>Cancel</code> existuje. Pokud nepotřebujete vykonávat nějaký údržbový či úklidový kód, jednoduše nastavte vlastnost <code>CancelDestinationPageUrl</code> – pak se o přesměrování automaticky postará samotný průvodce.
<code>FinishButtonClick</code>	Nastane, když se klikne na tlačítko <code>Finish</code> .
<code>NextButtonClick</code> <code>PreviousButtonClick</code>	Nastane, když se v některém z kroků klikne na tlačítko <code>Next</code> , resp. <code>Previous</code> . Protože ovšem existuje více než jedna možnost, jak se dostat z jednoho kroku na další, je lepší zpracovávat událost <code>ActiveStepChanged</code> .
<code>SideBarButtonClick</code>	Nastane, když se klikne na tlačítko v postranní nabídce odkazů.

Existují v podstatě dva programovací modely průvodce.

- **Potvrzovat průběžně.** To je rozumné, jestliže každý krok průvodce obaluje nedělitelnou (atomic-kou) nevratnou operaci. Pokud například máte do procesu zpracování objednávky zařazen krok pro autorizaci kreditní karty, za kterým následuje finální platba, nemůžete uživateli povolit, aby se vrátil zpět a upravil číslo kreditní karty. Tento model vytvoříte tak, že u některých (nebo u všech) kroků nastavíte vlastnost `AllowReturn` na `false` a změny budete potvrzovat v každém kroku jako reakci na událost `ActiveStepChanged`.
- **Potvrdit na konci.** Tento model je rozumný tehdy, když se v jednotlivých krocích získávají informace pro operaci, kterou je možné vykonat až na konci celého průvodce. Pokud shromažďujete informace o uživateli a hodláte pro něj vytvořit nový uživatelský účet teprve tehdy, až získáte všechny požadované informace, pravděpodobně uživateli dovolíte, aby se mohl v průběhu práce s průvodcem vracet a opravovat to, co uvedl v předchozích krocích. Kód pro vygenerování nového účtu spustíte až tehdy, až průvodce skončí, jako reakci na událost `FinishButtonClick`.

Chcete-li pro aktuální příklad implementovat model "potvrdit na konci", stačí reagovat na událost `FinishButtonClick`. Podívejte se na ukázkou, která vypíše shrnutí toho, co uživatel vybral v průvodci:

```
protected void Wizard1_FinishButtonClick
(object sender, WizardNavigationEventArgs e)
{
    StringBuilder sb = new StringBuilder();
    sb.Append("<b>You chose: <br />");
    sb.Append("Programming Language: ");
    sb.Append(lstLanguage.Text);
    sb.Append("<br />Total Employees: ");
    sb.Append(txtEmpCount.Text);
    sb.Append("<br />Total Locations: ");
    sb.Append(txtLocCount.Text);
    sb.Append("<br />Licenses Required: ");
    foreach (ListItem item in lstTools.Items)
    {
        if (item.Selected)
        {
            sb.Append(item.Text);
            sb.Append(" ");
        }
    }
    sb.Append("</b>");
    lblSummary.Text = sb.ToString();
}
```

Aby to celé fungovalo, musíte do posledního kroku přidat ovládací prvek `Label` s názvem `lblSummary`. V tomto příkladu se `lblSummary` umísťuje až ve finálním kroku, který zobrazuje shrnutí.

---

**TIP** Pokud chcete zjistit, kudy uživatel chodil po průvodci, pomůže vám metoda `Wizard.GetHistory()`, která vrací kolekci objektů `WizardStepBase`, jež už byly zpřístupněny, a které jsou chronologicky uspořádány v opačném pořadí. To znamená, že první prvek v kolekci reprezentuje předchozí krok, druhý prvek reprezentuje krok před předchozím krokem atd.

---

## Styly a šablony průvodce

Největší předností ovládacího prvku `Wizard` je nepochybně to, jak umožňuje přizpůsobovat vzhled. To znamená, že pokud chcete používat základní model (tzn. proces složený z několika kroků s navigačními tlačítky a různými událostmi), nejste přikováni k výchozímu uživatelskému rozhraní.

Máte k dispozici různé možnosti, které volíte podle toho, jak radikálně chcete průvodce změnit. U těch méně dramatických modifikací bude stačit nastavit pár vlastností nejvyšší úrovně. Můžete nastavovat barvu, písmo, rozstup či styl orámování (obdobně jako u jiných ovládacích prvků ASP.NET). Můžete také přizpůsobovat vzhled jednotlivých tlačítek. Pokud například chcete upravit tlačítko `Next`, můžete využít tyto vlastnosti: `StepNextButtonType` (použít tlačítko, odkaz, nebo obrázkový odkaz), `StepNextButtonText` (přizpůso-

bit text tlačítka nebo odkazu), `StepNextButtonImageUrl` (nastavit obrázek pro odkaz ve formě obrázku) a `StepNextButtonStyle` (použít styl ze stylového předpisu). Prostřednictvím vlastnosti `HeaderText` můžete také přidat vlastní záhlaví.

Větší kontroly nad průvodcem dosáhnete prostřednictvím stylů. S nimi můžete všelijak formátovat různé části průvodce. Styly používáte úplně stejně jako u bohatě vybavených datových ovládacích prvků, mezi něž patří `GridView`. Všechny styly, které můžete použít, jsou vypsané v tabulce 17-4. A podobně jako u jiných ovládacích prvků založených na stylech, i zde platí, že pokud jsou nastavení stylu konfliktní, konkrétnější nastavení stylu (jako je `SideBarStyle`) potlačí všeobecnější nastavení stylu (jako je `ControlStyle`). Například `StartNextButtonStyle` v prvním kroku potlačí nastavení `NavigationButtonStyle`.

**Tabulka 17-4. Styly ovládacího prvku Wizard.**

Styl	Popis
<code>ControlStyle</code>	Aplikuje se na všechny sekce ovládacího prvku Wizard.
<code>HeaderStyle</code>	Aplikuje se na sekci záhlaví ovládacího prvku Wizard, která je viditelná jen tehdy, přiřadíte-li text do vlastnosti <code>HeaderText</code> .
<code>SideBarStyle</code>	Aplikuje se na pruh odkazů ovládacího prvku Wizard.
<code>SideBarButtonStyle</code>	Aplikuje se jen na tlačítka v pruhu odkazů.
<code>StepStyle</code>	Aplikuje se na tu sekci ovládacího prvku, ve které definujete obsah daného kroku.
<code>NavigationStyle</code>	Aplikuje se na spodní oblast ovládacího prvku, kde se zobrazují navigační tlačítka.
<code>NavigationButtonStyle</code>	Aplikuje se jen na navigační tlačítka.
<code>StartNextButtonStyle</code>	Aplikuje se na tlačítko Next v prvním kroku (když je <code>StepType</code> nastavena na <code>Start</code> ).
<code>StepNextButtonStyle</code>	Aplikuje se na tlačítko Next v průběžných krocích (když je <code>StepType</code> nastavena na <code>Step</code> ).
<code>StepPreviousButtonStyle</code>	Aplikuje se na tlačítko Previous v průběžných krocích (když je <code>StepType</code> nastavena na <code>Step</code> ).
<code>FinishPreviousButtonStyle</code>	Aplikuje se na tlačítko Previous v posledním kroku (když je <code>StepType</code> nastavena na <code>Finish</code> ).
<code>CancelButtonStyle</code>	Aplikuje se na stornovací tlačítko, je-li vlastnost <code>Wizard.DisplayCancelButton</code> nastavena na <code>true</code> .

A konečně – pokud prostřednictvím vlastností a stylů nemůžete dosáhnout vámi požadované úrovně přizpůsobení, můžete ještě použít šablony, pomocí kterých můžete kompletně specifikovat vlastní vzhled ovládacího prvku Wizard. Při úpravách vzhledu prostřednictvím obvyklých technik dodáváte značkování pouze pro obsah kroku (jak jste to viděli na obrázku 17-1). Se šablonami můžete dodat značkování pro libovolný region průvodce, jako jsou záhlaví, pruh odkazů nebo tlačítka. Všechny šablony se deklarují odděleně od obsahu kroku. Na obrázku 17-7 vidíte místa, kde všude se dají specifikovat šablony.