KAPITOLA 17

Zdroje a lokalizace

Protože stále více a více firem proniká na mezinárodní trhy prostřednictvím internetu, je pro váš úspěch stále důležitější, aby vaše aplikace podporovaly různé kultury a .NET Framework má v sobě integrovanou infrastrukturu pro vytváření mezinárodních aplikací.

CLR v zásadě podporuje mechanismus pro balení a rozmisťování zdrojů s jakýmkoliv typem aplikace. CLR a knihovna základních tříd .NET Frameworku přichází s několika třídami pro správu a přístup ke zdrojům v aplikacích. Tyto třídy se nacházejí ve jmenných prostorech System.Resources a System.Globalization.

V této kapitole se dozvíte veškeré nezbytné informace o práci se zdroji v aplikacích ASP.NET, a také se více dozvíte o vytváření mezinárodních aplikací ASP.NET, které jsou založeny na vložených prostředcích a integrované lokalizační podpoře.

Zdroje v aplikacích .NET

V aplikacích se běžně používá řada obrázků a řetězců určených pro panely nástrojů, pro ikony v nabídkách, pro texty v nabídkách nebo pro různé nápisy. Měnit všechny řetězce a obrázky tohoto druhu může být strašná dřina, máte-li všechno přímo ve zdrojovém kódu. Aby bylo možné je v programu měnit co nejsnadněji, aniž by se musel procházet kompletní zdrojový kód a vyhledávat je v něm, dají se umístit do samostatných souborů. Pak se změny provádějí jenom na jediném místě.

.NET Framework a CLR nabízejí pro tento přístup skvělou podporu prostřednictvím vložených zdrojů (embedded resources). Všechny řetězce, obrázky i další typy dat, jednoduše všechno, co nemá být přímo ve zdrojovém kódu, se dá uložit do samostatných souborů zdrojů. Obvykle se tyto zdroje kompilují přímo do binární podoby samotné aplikace, takže se pak rozmisťují automaticky společně s aplikací, přičemž žádné rozmisťovací kroky navíc nejsou nutné.

Primárním účelem použití zdrojů je lokalizace. Pomocí zdrojů můžete definovat hodnoty vlastností ovládacích prvků (jako je text v ovládacím prvku Label) v různých souborech zdrojů — jeden soubor pro každou z kultur, kterou aplikace podporuje. Každý z těchto souborů zdrojů obsahuje řetězce (dvojice klíč/hodnota) lokalizovaných vlastností ovládacího prvku, které jsou přeloženy do odpovídající kultury. Při běhu pak CLR načte zdroje z patřičných souborů vložených zdrojů a použije je pro vlastnosti ovládacích prvků.

Zdroje jsou však prospěšné v mnoha ohledech, nejenom v oblasti lokalizace. V aplikacích Windows se zdroje používají pro ikony panelů nástrojů, pro ikony v nabídkách a pro stavové ikony, takže se pak nemusí roz-

misťovat samostatně s aplikací (což webových aplikací není příliš běžné). Zdroje se dají také využít ve vašich vlastních třídách instalátorů pro vkládání dodatečných rozmisťovacích skriptů (jako jsou skripty pro vytváření databází nebo modifikace katalogu COM+), takže je nemusíte v rámci rozmisťovacího procesu rozmisťovat samostatně.

Definice zdrojů jsou obvykle uloženy v souborech .resx. Jsou to jednoduše soubory XML obsahující buď hodnoty řetězce, nebo odkazy na externí soubory. Tyto řetězce a odkazované soubory se pak zkompilují jako vložené zdroje do binární podoby aplikace. Zdroje můžete ve Visual Studiu přidávat do projektu tak, že vyberete Add > Assembly Resource File.

Následující příklad ukazuje výtah z ukázkového souboru .resx. Soubor obsahuje jednoduché řetězce a odkaz na externí soubor obrázku.

```
<root>
  <resheader name="resmimetype">
    <value>text/microsoft-resx</value>
  </resheader>
  <resheader name="version">
    <value>2.0</value>
  </resheader>
  ...
  <data name="Binary Code Sm"
  type="System.Resources.ResXFileRef, System.Windows.Forms">
    <value>
      Binary Code Sm.png;System.Drawing.Bitmap, System.Drawing,
      Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a
    </value>
  </data>
  <data name="LegendAge">
    <value xml:space="preserve">Age</value>
  </data>
  <data name="LegendFirstname">
    <value xml:space="preserve">Firstname</value>
  </data>
  <data name="LegendLastname">
    <value xml:space="preserve">Lastname</value>
  </data>
</root>
```

POZNÁMKA Tohle je jenom stručný výtah souboru .resx. Visual Studio totiž automaticky generuje celé hejno komentářů a schémat XML pro soubor zdrojů v jediném XML souboru.

Existuje ovšem i jiný způsob, jak v aplikaci zkompilovat soubor jako vložený zdroj. Přidejte soubor do projektu a v okně Properties pro něj nastavte vlastnost BuildAction na Embedded Resource. To ale funguje jenom pro projekty Windows, jako jsou knihovny tříd, služby Windows, nebo pro aplikace formulářů Win-

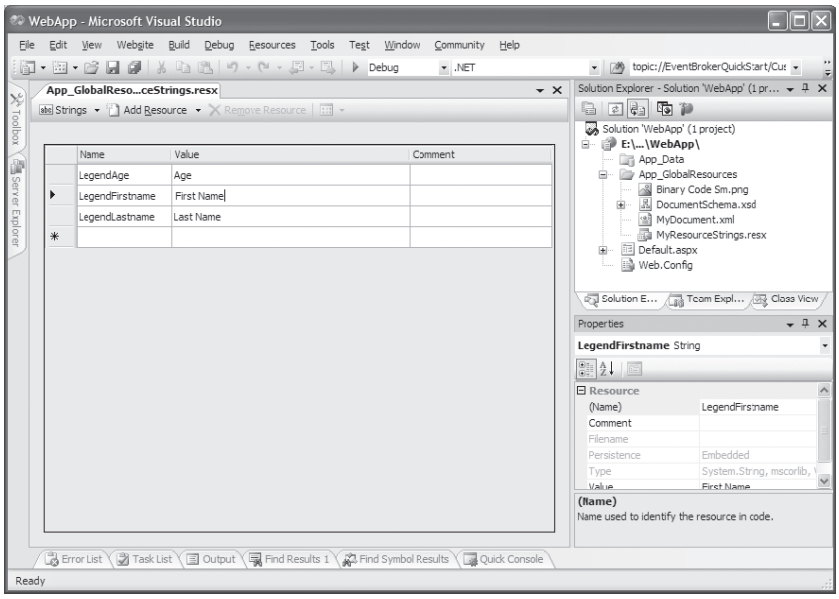
dows. U aplikací ASP.NET musíte použít přístup popsaný v úvodu kapitoly. Ve všech případech však můžete ke zdrojům přistupovat programátorsky pomocí třídy ResourceManager.

POZNÁMKA Zdroje můžete do aplikace přidávat také jako textové soubory. Takové textové soubory se skládají z dvojic klíč/hodnota, na každém řádku je pak jedna dvojice ve formátu klíč = hodnota. Tyto soubory se pak dají zkompilovat do binárního formátu s příponou souboru .resources pomocí nástroje resgen.exe . Takové soubory .resources pak můžete v aplikaci použít jako vložené zdroje (BuildAction=Embedded--Resource). Protože se však jedná o starší způsob správy zdrojů, není to už doporučováno.

V průběhu práce s následující ukázkovou webovou aplikací se dozvíte, jak se používají zdroje pro nastavování vlastností ovládacích prvků i pro získávání jiných informací, jako jsou třeba šablony dokumentu pro generování sestav. Webová aplikace od uživatele požaduje, aby zadal své křestní jméno, příjmení a věk. Na základě těchto informací aplikace vygeneruje jednoduchý dokument Microsoft Wordu. Pro tento účel se šablona dokumentu Microsoft Wordu, který se má vygenerovat, sváže se schématem XML, uloží do formátu XML a přidá jako vložený zdroj.

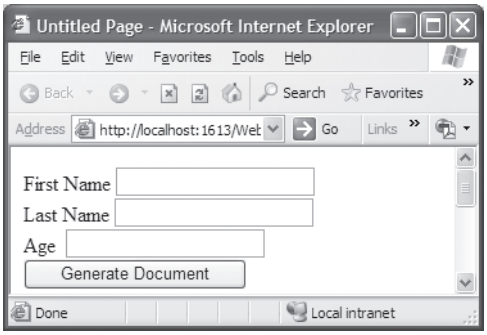
POZNÁMKA Schopnosti XML produktu Microsoft Office 2003, které uvádíme v této kapitole, vyžadují, aby byl u klienta (nikoliv na serveru) nainstalovaný Microsoft Word 2003 Professional Edition. Nemáte-li Microsoft Word 2003, můžete stejné věci otestovat pomocí vašich vlastních souborů šablon. Chcete-li se dozvědět více podrobností o Microsoft Office 2003 a XML, navštivte stránku `http://msdn.microsoft.com/office`.

Řetězce, a také dokument Microsoft Wordu používaný v této aplikaci, budou vloženy jako zdroje. Poté, co přidáte assembly souboru zdroje do projektu, Visual Studio nabídne editor zdrojů, v němž můžete upravovat obsah souboru .resx. Můžete v něm do projektu přidávat řetězce nebo jiný typ souboru, viz obrázek 17-1.



Obrázek 17-1. *Editor zdrojů Visual Studia 2005.*

Prostřednictvím Add Resource > Add Existing File můžete do projektu přidat jakýkoliv externí soubor uložený v jakémkoliv formátu na pevném disku. Takové soubory se automaticky přidají do jednoho z adresářů pro zdroje webu ASP.NET a zkompilují se jako vložené zdroje do výsledné binární podoby aplikace. Visual Studio a ASP.NET rozlišují globální zdroje a lokální zdroje. Globální zdroje jsou přístupné zevnitř jakékoliv stránky aplikace, lokální zdroje se vztahují ke stránce a jsou přístupné jenom zevnitř této stránky. A dále – soubory .resx přidané do složky App_GlobalResources jsou přístupné ze všech stránek aplikace. Momentální rozhraní aplikace, která běží bez využití vložených zdrojů, vidíte na obrázku 17-2.



Obrázek 17-2. *Běžící ukázková aplikace, která nepoužívá třídu ResourceManager.*

Jak je z obrázku 17-2 vidět, titulky ovládacích prvků Label nejsou inicializovány. Nyní přidáme kód, ve kterém budeme inicializovat vlastnosti textu, a také vygenerujeme jednoduchý dokument. V průběhu práce s příkladem se naučíte různé způsoby, jak se dá přistupovat k vloženým zdrojům. Kromě toho ještě uvidíte, že zdroje se dají používat k různým účelům, přestože se nejčastěji používají pro potřeby lokalizace.

V zásadě se dá ke vloženým zdrojům přistupovat přes třídu, kterou vygeneruje Visual Studio. Tato třída se vygeneruje na základě informací uložených v souboru .resx. To znamená, že když vytvoříte nebo modifikujete zdroje v editoru zdrojů (viděli jste ho na obrázku 17-1), Visual Studio vloží zdroje do aplikace, a za scénou automaticky vytvoří silně typovanou třídu pro přístup k těmto zdrojům prostřednictvím vlastností (názvy vlastností se odvodí z názvů zdrojů, které jste si vybrali v editoru zdrojů). Takže – každá položka v souboru zdrojů bude směřovat na nějakou vlastnost vygenerované třídy; název třídy bude odvozen z názvu souboru zdrojů, který jste vytvořili.

Následující fragment kódu předvádí, jak se dá přistupovat k zdrojům, které jsou uloženy v předtím přidaném souboru MyResourceStrings.resx. Název třídy silně typované třídy zdrojů tudíž bude MyResourceStrings, přičemž třída bude obsahovat tři vlastnosti pro zdroje, které jsme specifikovali dříve.

```
protected void Page_Load(object sender, EventArgs e)
{
  // Tohle jsou jednoduché řetězcové zdroje
  LegendFirstname.Text = Resources.MyResourceStrings.LegendFirstname;
  LegendLastname.Text = Resources.MyResourceStrings.LegendLastname;
  LegendAge.Text = Resources.MyResourceStrings.LegendAge;
  // Tohle je dokument XML, který jsme do zdrojů přidali jako soubor
  DocumentXml.DocumentContent = Resources.MyResourceStrings.MyDocument;
}
```

POZNÁMKA V oddílu "Lokalizace webových aplikací" později uvidíte, že Visual Studio 2005 a ASP.NET 2.0 poskytují mnohem lepší podporu pro lokalizaci titulků a jiných vlastností ovládacích prvků. V této části kapitoly chceme pouze ukázat nízkoúrovňové API pro správu jakéhokoliv typu zdrojů, nikoliv pouze zdrojů sloužících potřebám lokalizace.

Vygenerovaná třída interně používá instanci třídy ResourceManager, která je definována ve jmenném prostoru System.Resources. Instance této třídy je přístupná prostřednictvím vlastnosti ResourceManager vygenerované třídy. Procedury vlastností pro přístup k samotným vloženým zdrojům jsou interně pouhé obaly okolo volání jedné z metod GetXxx (tj. GetString nebo GetStream) této instance ResourceManager. Například zdroj, který je přístupný prostřednictvím vygenerované procedury vlastnosti Resources.MyResourceStrings. LegendAge, je také přístupný prostřednictvím ResourceManager.GetString("LegendAge"), jak je to vidět na obrázku 17-3.

Obrázek 17-3. *Metody třídy ResourceManager.*

Vlastnost ResourceManager vygenerované třídy vytvoří instanci manažera zdrojů automaticky, jak to ukazuje následující fragment kódu. Samozřejmě, tohle může udělat kdokoliv.

```
ResourceManager ResMgr = new ResourceManager("Resources.MyResourceStrings",
Assembly.GetAssembly(typeof(Resources.MyResourceStrings)));
```

První parametr je základní název (base name) zdrojů, které má třída ResourceManager načíst. Druhý parametr specifikuje assembly, do kterého se zdroje zkompilovaly. Jestliže se zdroje zkompilují do assembly, který vykonává tento kód, je postačující zavolat metodu GetExecutingAssembly() třídy System.Reflection. Assembly. Většinou to platí pro knihovny tříd nebo klasické aplikace Windows. V ASP.NET je to ale trochu jinak, protože generuje a kompiluje assembly automaticky. Infrastruktura ASP.NET ve skutečnosti interně vytvoří různé assembly pro kód stránky, a pro globální zdroje. A název dynamicky generované assembly je v zásadě neznámý, protože i ten určuje infrastruktura. Proto to s metodou GetExecutingAssembly() nepůjde. Alternativu nabízí metoda GetAssembly s popisem typu generované třídy zdrojů. S ní se dá totiž vytvořit vlastní instance třídy ResourceManager, protože tento typ sídlí v assembly, kterou vytvoří ASP.NET pro vložené zdroje obsažené v této třídě.

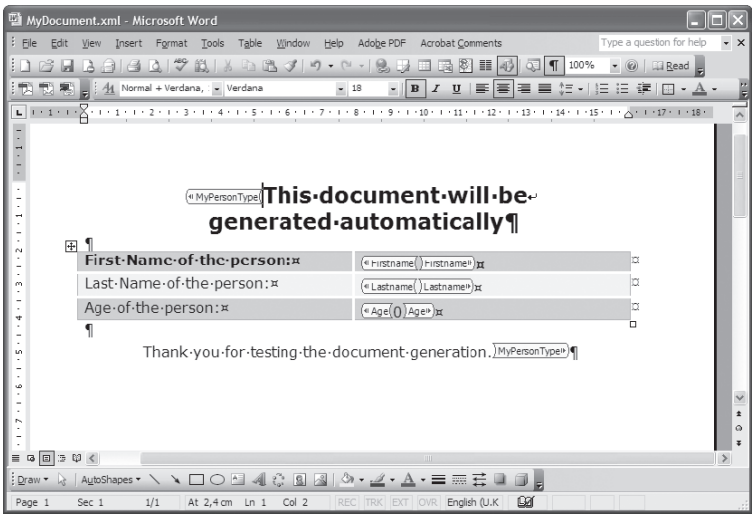
Když už víte, jak se používají zdroje, seznamte se s kódem, jímž se vygeneruje dokument Microsoft Wordu, když uživatel klikne na stránce na tlačítko. Pro tento účel zužitkujeme funkcionalitu XML Microsoft Wordu 2003. Postupujte takto:

1. Nejprve vytvořte dokument Microsoft Wordu jako obvykle a uložte ho jako dokument XML, nikoliv v binárním formátu DOC. Formát vyberte v seznamu File Types (Typ souboru) dialogu Save as (Uložit jako) Microsoft Wordu.

2. Teď vytvořte schéma XML. Toto schéma XML musíte připojit k dokumentu Microsoft Wordu prostřednictvím podokna úloh XML Structure (Struktura XML). Do podokna úloh se dostanete tak, že vyberete View > Task Panel (Zobrazit > Podokno úloh), a pak kliknete v titulku podokna. Z nabídky vyberte XML Structure (Struktura XML). Podokno úloh umožňuje přidat schéma XML do dokumentu a přiřadit prvky schématu XML sekcím dokumentu Microsoft Wordu. Aplikace použije prvky definované ve schématu XML pro vyhledání patřičných sekcí v dokumentu, jejichž obsah vygeneruje prostřednictvím System.Xml. (Podrobnosti o System.Xml viz kapitola 12.)

3. Nakonec přidejte do svého řešení dokument Wordu ve formátu XML jako zdroj, pomocí výše zmíněného editoru zdrojů (viz obrázek 17-1). Schéma XML, které máte vytvořit ve svém řešení, je vypsané níže. Až schéma vytvoříte, vytvořte dokument Wordu a svažte schéma s dokumentem prostřednictvím úlohy XML Structure (Struktura XML). Dokument Wordu se sdruženými prvky XML vidíte na obrázku 17-4.

```
<xs:schema id="DocumentSchema"
     targetNamespace="uri:AspNetPro20/Chapter17/Demo1"
     elementFormDefault="qualified" xmlns="uri:AspNetPro20/Chapter17/Demo1"
      xmlns:mstns="uri:AspNetPro20/Chapter17/Demo1"
      xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="MyPersonType">
    <xs:complexType mixed="true">
      <xs:all>
        <xs:element name="Firstname" type="xs:string" />
        <xs:element name="Lastname" type="xs:string" />
        <xs:element name="Age" type="xs:int" />
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```



Obrázek 17-4. *Dokument Microsoft Wordu spojený se schématem XML.*

Chcete-li vygenerovat dokument Microsoft Wordu na základě šablony XML vytvořené výše, není k tomu potřeba nic víc, než ji načíst z vloženého zdroje do instance XmlDocument. (Další informace o XML najdete v kapitole 12.) Podíváte-li se pozorněji na dokument Microsoft Wordu, který jste uložili jako XML, uvidíte, že prvky XML definované ve schématu jsou přímo začleněny do prvků XML, které definují formátování a obsah dokumentu Microsoft Wordu. Následující výpis ukazuje výtah z dokumentu Microsoft Wordu, který byl uložen jako soubor XML (a vložen do aplikace jako zdroj):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<?mso-application progid="Word.Document"?>
<w:wordDocument xmlns:w="http://schemas.microsoft.com/office/word/2003/wordml"
...
  xmlns:ns1="uri:AspNetPro20/Chapter17/Demo1">
  <o:DocumentProperties>
    <o:Title>This document will automatically be generated:</o:Title>
    <o:Author>VB.NET Trainings</o:Author>
    <o:LastAuthor>VB.NET Trainings</o:LastAuthor>
    <o:Revision>8</o:Revision>
    <o:TotalTime>0</o:TotalTime>
...
...
  <w:body>
    <wx:sect>
      <ns1:MyPersonType>
        <w:p>
          <w:r>
            <w:rPr>
              <w:rFonts w:ascii="Verdana" w:h-ansi="Verdana" />
              <wx:font wx:val="Verdana" />
              <w:b />
              <w:sz w:val="36" />
              <w:sz-cs w:val="36" />
              <w:lang w:val="EN-GB" />
            </w:rPr>
            <w:t>This document will be</w:t>
          </w:r>
...
...
            <ns1:Firstname>
              <w:tc>
                <w:tcPr>
                  <w:tcW w:w="4606" w:type="dxa" />
                </w:tcPr>
                <w:p>
                  ...
                </w:p>
              </w:tc>
```

```
            </ns1:Firstname>
...
      </ns1:MyPersonType>
    </wx:sect>
  </w:body>
</w:wordDocument>
```

Jak vidíte, prvky definované ve schématu XML vytvořeném výše se zařadí přímo do souboru XML Microsoft Wordu. Všechny ostatní značky definují formátování a další volby, z nichž se skládá kompletní dokument Microsoft Wordu. To znamená, že všechny věci, které byly dříve zapouzdřeny v binárním formátu dokumentů Microsoft Wordu (formát DOC), jsou nyní také dostupné jako XML. Proto je zpracování dokumentů Microsoft Wordu z webové aplikace na straně serveru právě tak snadné, jako zpracování souborů XML. Schéma XML, které definuje XML prvky Microsoft Wordu, je dobře zdokumentováno a dá se stáhnout (včetně dokumentace) z `http://msdn.microsoft.com/office`. Co se týče našeho příkladu, musíte vědět ještě dvě věci. Zaprvé, dokument Microsoft Wordu můžete načíst do instance XmlDocument, a pak měnit jeho obsah prostřednictvím jednoduchých operací XML, mezi něž patří klonování uzlů a přidávání nových uzlů. Zadruhé, musíte znát význam jedné speciální značky XML – je to značka <w:r />, která se nazývá jako word run. Představuje takový XML fragment dokumentu Microsoft Wordu, který zapouzdřuje jednoduchý text uvnitř odstavce. A proto – chcete-li programátorsky dostat nějaký text na konkrétní pozici v XML dokumentu Microsoft Wordu, musíte absolvovat následující čtyři kroky:

1. Otevřete dokument Microsoft Wordu pomocí System.Xml.XmlDocument.
2. Vykonejte dotaz XPath, kterým vyhledáte nějaký prvek vašeho vlastního schématu XML – například prvek ns1:Firstname v dokumentu předvedeném výše.
3. Pak vykonejte dotaz XPath na uzel vrácený z předchozího dotazu XPath, pomocí kterého naleznete pod vašim vlastním uzlem XML (ns1:Firstname) další značku word run (<w:r />).
4. A nakonec můžete do značky <w:r> přidat značku <w:t> (Word text) obsahující text, který chcete do dokumentu Microsoft Wordu přidat.

Následující kód ukazuje, jak se otevře dokument Microsoft Wordu se třídou XmlDocument. Pak se vykoná dotaz XPath, který v dokumentu najde prvek ns1:Firstname. Poté, co prvek najde, změní se InnerXml na word run obsahující text, který uživatel zadal do ovládacího prvku TextBox ASP.NET. Pak kód udělá totéž pro sekce ns1:Lastname a ns1:Age.

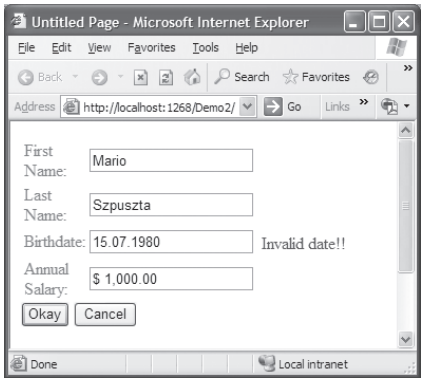
```
protected void GenerateAction_Click(object sender, EventArgs e)
{
  XmlDocument doc = new XmlDocument();
  doc.LoadXml(Resources.MyResourceStrings.MyDocument);
  XmlNode TextNode;
  XmlNamespaceManager NsMgr = new XmlNamespaceManager(doc.NameTable);
  NsMgr.AddNamespace("ns1", "uri:AspNetPro20/Chapter17/Demo1");
  NsMgr.AddNamespace("w",
  "http://schemas.microsoft.com/office/word/2003/wordml");
  TextNode = doc.SelectSingleNode("//ns1:Firstname//w:p", NsMgr);
  TextNode.InnerXml = string.Format("<w:r><w:t>{0}</w:t></w:r>",
  TextFirstname.Text);
```

```
  TextNode = doc.SelectSingleNode("//ns1:Lastname//w:p", NsMgr);
  TextNode.InnerXml = string.Format("<w:r><w:t>{0}</w:t></w:r>",
  TextLastname.Text);
  TextNode = doc.SelectSingleNode("//ns1:Age//w:p", NsMgr);
  TextNode.InnerXml = string.Format("<w:r><w:t>{0}</w:t></w:r>",
  TextAge.Text);
  // Vyčistí odpověď
  Response.Clear();
  Response.ContentType = "application/msword";
  Response.Write(doc.OuterXml);
  Response.End();
}
```

Protože byl do aplikace dodán jako vložený zdroj samotný XML dokument Microsoft Wordu, můžete ho získat prostřednictvím vygenerované třídy zdrojů, jako jsme to udělali v předchozí ukázce kódu. A ještě připomeňme, že řetězcová šablona <w:r><w:t>{0}</w:r></w:t>, kterou jsme použili pro přidání textu do dokumentu Wordu, je perfektním kandidátem na to, aby byla sama také vložena jako zdroj.

Lokalizace webových aplikací

Infrastruktura prezentovaná v první části této kapitoly poskytuje nutný základ pro lokalizaci jakéhokoliv typu aplikace založené na .NET, včetně aplikací Windows, knihoven tříd, služeb a (samozřejmě) webových aplikací. Než se začnete seznamovat s odbornými detaily lokalizace aplikací, prodiskutujeme hlavní výzvy, které nám lokalizace předkládá. Několik lokalizačních výzev webové aplikace vidíte na obrázku 17-5.



Obrázek 17-5. *Ukázková aplikace předvádí, co by se mělo lokalizovat.*

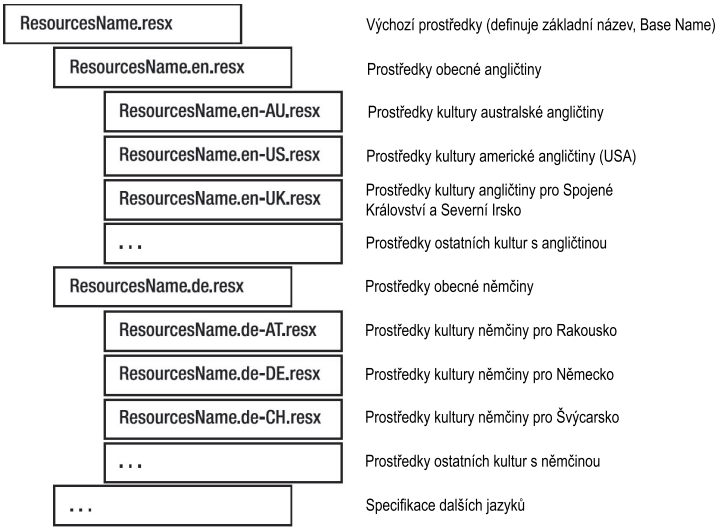
Přestože je to velmi jednoduchá aplikace, předvádí hlavní výzvy, jimž čelíte při lokalizaci webových aplikací. Nejjednodušší výzvou jsou samozřejmě řetězce textu v popiscích a na tlačítkách. Musíte také lokalizovat formát hodnot vyjadřujících peněžní částky, datum a čas. V těchto situacích musíte také lokalizovat ověřovací výraz validačního ovládacího prvku. A konečně, když v kódu provádíte rozklad (parse) informací, abyste je pak mohli uložit do databáze, nebo je mohli dále zpracovat, musíte brát také v úvahu formát data, času a peněžních částek. Všechny tyto hodnoty mohou být různé, a proto se musejí přizpůsobovat pro jednotlivé kultury.

Lokalizace a společný runtime jazyků (CLR)

Obvykle se zdroje vytvářejí pro každou kulturu, kterou má aplikace podporovat. V předchozím příkladu jste se dozvěděli, jak se vytvoří zdroje pro výchozí kulturu, protože jsme neuvedli žádné informace, které by byly specifické pro nějakou konkrétní kulturu. Tyto zdroje použije CLR při lokalizaci jako zdroj poslední záchrany.

CLR definuje chování pro hledání zdrojů, které jsou specifické pro jednotlivé kultury. Říká, že každá sada zdrojů má definovat základní název (base name), který se specifikuje prostřednictvím první části názvu souboru zdrojů. Druhá část je název, který jsme v předchozím příkladu této kapitoly neuvedli, a který definuje kulturu. Jestliže část názvu definující kulturu není uvedena, použijí se zdroje definované v tomto souboru zdrojů jako výchozí zdroje. Pokud je například základní název souboru vložených zdrojů MyResourceStrings.resx, pak název specifický pro kulturu americké angličtiny bude MyResourceStrings.en-US.resx.

CLR definuje hierarchii kultur a jazyků, jejíž nástin vidíte na obrázku 17-6. (Kompletní seznam kultur a jejich hierarchií naleznete ve třídě CultureInfo v dokumentaci MSDN.)



Obrázek 17-6. *Hierarchie kultur, jak ji definuje CLR.*

CLR hledá zdroje specifické pro jednotlivé kultury automaticky, na základě jejich hierarchií. V zásadě to funguje tak, že se pokouší najít takovou sadu zdrojů, která se co nejvíce shoduje s aktuálně nastavenou kulturou. Obecnější zdroje vzhledem ke kulturám se vždy používají jako východisko z nouze, nejsou-li k dispozici zdroje pro danou konkrétní kulturu. A dále – jestliže nejsou pro aktuální kulturu vloženy do aplikace žádné zdroje specifické pro tuto kulturu, CLR automaticky použije výchozí zdroje vložené do aplikace. Jestliže výchozí zdroje neobsahují hodnotu pro požadovaný zdroj, CLR vygeneruje výjimku.

Načítáte-li zdroje prostřednictvím třídy ResourceManager, vždy uvádíte pouze základní název. Pokud jste tedy do projektu přidali soubory zdrojů s názvy MyResourceStrings.resx a MyResourceStrings.en-US.resx, při vytváření instance třídy ResourceManager uvedete jenom základní název, bez ohledu na to, jaká kultura se bude používat. Například:

```
ResourceManager ResMgr = new ResourceManager("Resources.MyResourceStrings",
Assembly.GetAssembly(typeof(Resources.MyResourceStrings)));
```

POZNÁMKA ASP.NET automaticky přidává k názvům zdrojů prefix Resources. Takže základní název globálních zdrojů přidávaných do projektu webové aplikace má vždy formát Resources.NazevSouboruMehoZdroje. Když přidáváte zdroje například do nějaké knihovny tříd, můžete prefix Resources vynechat.

Uvádět soubory zdrojů, které jsou specifické pro nějaké kultury, není povinné. CLR umí načíst zdroje specifické pro danou kulturu ze souborů zdrojů založených na "aktuální" kultuře. Ale co to vlastně je aktuální kultura? HTTP specifikuje záhlaví HTTP, což umožňuje prohlížeči, aby odeslal z klienta na server informace vztahující se ke kultuře. Na základě této informace pak můžete vytvořit instanci třídy CultureInfo, což předvedeme v příštím oddílu.

Třída CultureInfo

Všechny informace, které jsou specifické vzhledem ke kulturám, se ukládají jako instance třídy CultureInfo. Tato třída je definována ve jmenném prostoru System.Globalization. Umožňuje se dotazovat na informace související s kulturou, jako jsou název kultury, její formát data, její formát pro čísla, nebo formát pro peněžní částky. Následující kód vytvoří instanci CultureInfo na základě jazyka, který odeslal prohlížeč:

```
protected void Page_Load(object sender, EventArgs e)
{
  CultureInfo ci;
  if ((Request.UserLanguages != null) && (Request.UserLanguages.Length > 0))
  {
    ci = new CultureInfo(Request.UserLanguages[0]);
    System.Threading.Thread.CurrentThread.CurrentUICulture = ci;
  }
  else
  {
    ci = System.Threading.Thread.CurrentThread.CurrentUICulture;
  }
  StringBuilder MessageBuilder = new StringBuilder();
  MessageBuilder.Append("Current culture info: ");
  MessageBuilder.Append("<BR>");
  MessageBuilder.AppendFormat("-) Name: {0}", ci.Name);
  MessageBuilder.Append("<BR>");
  MessageBuilder.AppendFormat("-) ISO Name: {0}",
  ci.ThreeLetterISOLanguageName);
  MessageBuilder.Append("<BR>");
  MessageBuilder.Append("-) Currency Symbol: " +
  ci.NumberFormat.CurrencySymbol);
  MessageBuilder.Append("<BR>"); MessageBuilder.Append("-) Long
  Date Pattern: " + ci.DateTimeFormat.LongDatePattern);
  LegendCI.Text = MessageBuilder.ToString();
}
```

Objekt Request má vlastnost, která se jmenuje UserLanguages. Obsahuje informaci o jazyku, kterou odeslal prohlížeč prostřednictvím HTTP záhlaví na server. Odesílání informace o kultuře prostřednictvím záhlaví

HTTP nicméně není povinné, takže webová aplikace nemusí tuto informaci získat. V takovém případě specifikujte výchozí kulturu v souboru web.config:

```
<globalization enableClientBasedCulture="true" culture="de-DE" uiCulture="de-DE"/>
```

Nastavení specifikovaná v tomto konfiguračním prvku se automaticky nastaví do vlastností Thread.CurrentThread.CurrentUICulture a Thread.CurrentThread.CurrentCulture. CLR pak tyto vlastnosti použije nejenom pro nalezení patřičných data ve vložených prostředcích, ale také pro formátovací metody – jako jsou ToString() nebo Parse() – pro řízení různých, na kultuře závislých, typů (jako jsou DateTime nebo Decimal).

Jestliže prohlížeč odešle z klienta informaci o kultuře, musíte na začátku ovladače události Page_Load překrýt tato nastavení, jak jsme to předvedli v předchozím příkladu. Vlastnost CurrentCulture ovlivňuje chování formátovacích funkcí. Pokud například zavoláte DateTime.Now.ToString nebo DateTime.Now.Parse, bude použit formát data založený na vlastnosti CurrentCulture vlákna. Totéž platí pro formáty čísel.

ResourceManager oproti tomu použije (při hledání zdrojů specifických vzhledem ke kultuře) CurrentUICulture. To znamená, že když se obdrží požadavek od prohlížeče, ASP.NET automaticky inicializuje vlastnost CurrentUICulture instance System.Threading.Thread.CurrentThread. Na základě této vlastnosti CurrentUICulture, a také na základě hierarchie kultur, kterou definuje CLR (viděli jste ji na obrázku 17-6), třída ResourceManager automaticky získá lokalizované zdroje z patřičných vložených zdrojů, když se volá některá z metod GetXxx. Několik příkladů tohoto chování předvádí tabulka 17-1, přičemž se předpokládá, že aplikace je vybavena následujícími soubory zdrojů:

- **MyResources.resx**. Výchozí zdroje s hodnotami pro Firstname, Lastname, Age a DocumentName.
- **MyResources.en.resx**. Výchozí zdroje anglických kultur s hodnotami pro Firstname, Lastname a Age.
- **MyResources.de.resx**. Výchozí zdroje německých kultur s hodnotami pro Firstname, Lastname a Age.
- **MyResources.de-DE.resx**. Zdroje německé kultury pro Německo s hodnotami pro Firstname a Lastname.
- **MyResources.de-AT.resx**. Zdroje německé kultury pro Rakousko s hodnotami pro Firstname a Lastname.

Tabulka 17-1. Ukázky chování manažera zdrojů v různých situacích.

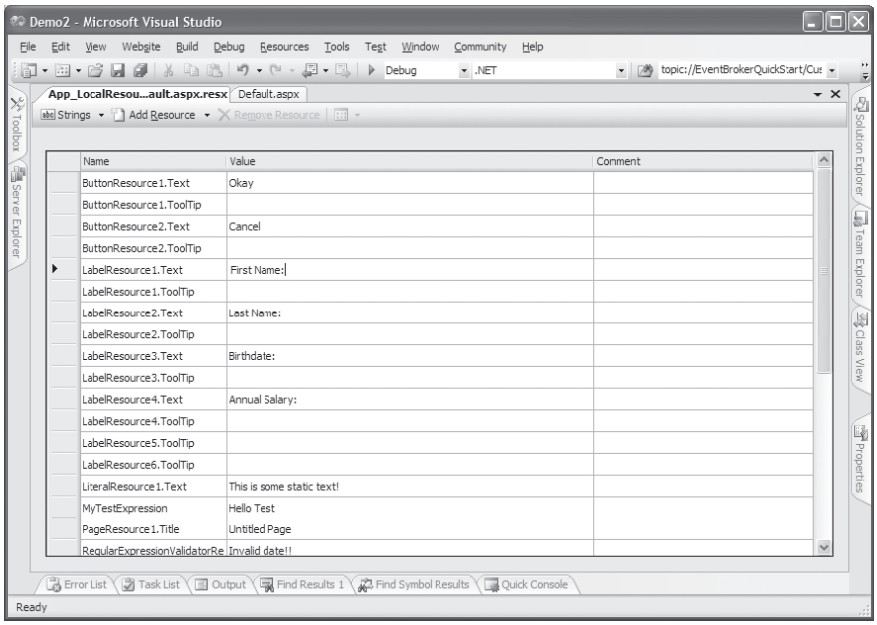
CurrentUICulture	Volání metody	Výsledek
en-US	GetString("Firstname")	Použije se hodnota Firstname z MyResources.en.resx, protože v aplikaci neexistují zdroje kultury pro americkou angličtina.
en-UK	GetString("Firstname")	Použije se hodnota Firstname z MyResources.en.resx, protože v aplikaci neexistují zdroje kultury pro angličtinu Velké Británie a Severního Irska.
en-US	GetString("DocName")	Použije se hodnota DocumentName z MyResources.resx, protože v souboru zdrojů není specifikována hodnota pro klíč DocName pro kulturu angličtiny.

CurrentUICulture	Volání metody	Výsledek
de-DE	GetString("Firstname")	Použije se hodnota Firstname z MyResources.de--DE.resx.
de-DE	GetString("Age")	Použije se hodnota Age z MyResources.de.resx, protože tato hodnota není specifikována v souboru MyResources.de-DE.resx.
de-AT	GetString("Lastname")	Použije se hodnota Lastname z MyResources.de-AT.resx.
de-AT	GetString("DocumentName")	Použije se hodnota DocumentName z MyResources.resx, protože tato hodnota neexistuje ani v MyResources.de-AT.resx, ani v MyResources.de.resx.

Lokální zdroje pro jedinou stránku

Třídy, které jste doposud viděli, poskytují základní infrastrukturu pro lokalizaci jakéhokoliv typu aplikace založené na .NET. S původním ASP.NET 1.x jste museli pomocí této infrastruktury lokalizovat obsah ovládacích prvků ručně.

To se naštěstí s vydáním ASP.NET 2.0 a Visual Studia 2005 změnilo, takže nyní mají weboví vývojáři při lokalizacích podporu od samého začátku a ve stejném duchu, jak to je u formulářů Windows. Chcete-li lokalizovat nějakou stránku, vyberte jednoduše Tools > Generate Local Resources. Visual Studio vygeneruje soubor zdrojů ve složce App_LocalResources, který bude obsahovat hodnoty všech ovládacích prvků té stránky, která je právě otevřena v době návrhu. Vygenerované zdroje pro předchozí příklad vidíte na obrázku 17-7.



Obrázek 17-7. *Vygenerované zdroje pro aplikaci z obrázku 17-5.*

Visual Studio vygeneruje zdroje pro několik vlastností každého ovládacího prvku. Zdroje vždy dostanou název ovládacího prvku jako prefix, a jako sufix název vlastnosti. Visual Studio automaticky vygeneruje pro ovládací prvky stránky pouze výchozí zdroje. Další zdroje specifické pro jednotlivé kultury musíte přidat ručně tak, že zkopírujete vygenerované zdroje a udělíte jim patřičný název (například Default.aspx.en-US.resx).

Nástroje pro generování zdrojů vytvoří položku pro každou vlastnost, která je v ovládacím prvku označena atributem [Localizable]. Chcete-li tedy vytvořit nějaký vlastní, lokalizovatelný ovládací prvek, musíte označit atributem [Localizable] všechny vlastnosti, které mají být lokalizovatelné, jako zde:

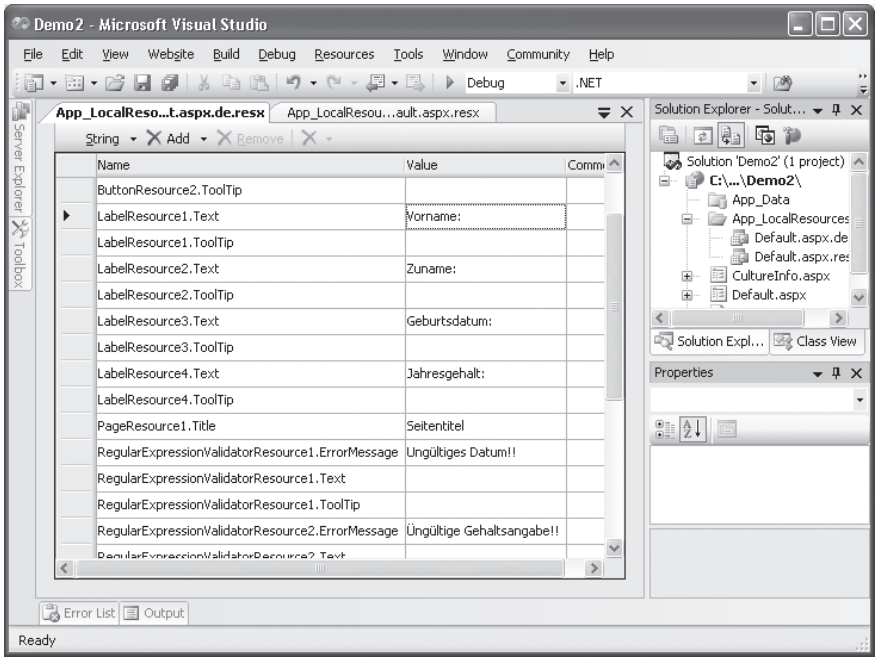
```
[Localizable(true)]
public string MyProperty
{
  get { ... }
  set { ... }
}
```

Když zkopírujete zdroje vygenerované výše a přejmenujete kopii na Default.aspx.de.resx, přidáte tím do aplikace zdroje, kterou budou specifické pro kulturu němčiny. Runtime pak bude schopno inicializovat vlastnosti ovládacího prvku na základě vlákna CurrentUICulture pomocí řetězců, které jsou uloženy v souboru vložených zdrojů pro tuto kulturu. Adaptovaný soubor zdrojů vidíte na obrázku 17-8, na dalším obrázku 17-9 pak vidíte běžící aplikaci, ve které je aktuální kulturou němčina, a konečně na obrázku 17-10 pak vidíte běžící aplikaci, ve které je aktuální kulturou angličtina.

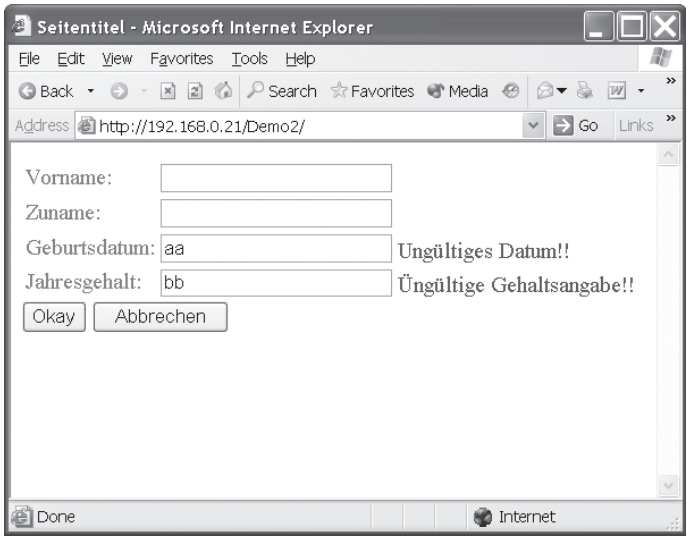
Kromě toho, že Visual Studio vygenerovalo soubor zdrojů, změnilo také zdrojový kód stránky. Pro každou lokalizovatelnou vlastnost (s atributem [Localizable]) každého ovládacího prvku umístěného na stránce byl přidán lokalizační výraz, jak to ukazuje následující fragment kódu:

```
<asp:Label ID="LegendFirstname" runat="server"
Text="Firstname:" meta:resourcekey="LabelResource1"></asp:Label>
```

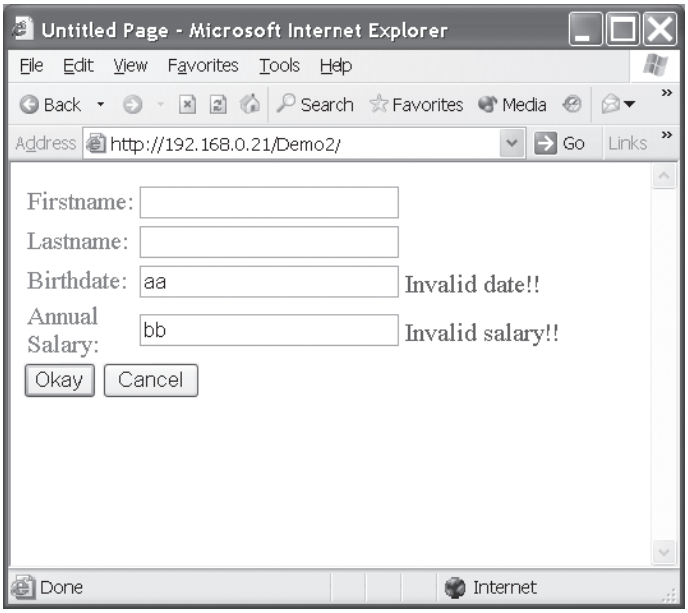
Lokalizační výrazy se specifikují pomocí atributu meta:resourceKey dané značky. V průběhu procesu rozkladu (parse) stránky projde runtime ovládacími prvky a vygeneruje nezbytný kód pro získávání zdrojů prostřednictvím třídy ResourceManager. Vlastnosti přiřazené deklarativně zůstanou netknuté a zobrazují se v režimu návrhu.



Obrázek 17-8. *Přidaný soubor zdrojů pro jinou kulturu.*



Obrázek 17-9. *Aplikace běžící na systému s místními nastaveními pro němčinu.*



Obrázek 17-10. *Aplikace běžící na systému s místními nastaveními pro angličtinu.*

Lokalizační výraz uvedený v předchozím fragmentu kódu je tzv. implicitní lokalizační výraz. Implicitní lokalizační výrazy jsou něco jako zkratky ke klíčům zdrojů uvedených ve vložených zdrojích pro stránku. Měly by respektovat jmenné konvence, které používá Visual Studio při generování zdrojů (například Kliczdroje. Nazevvlastnosti). Implicitní lokalizační výrazy jednoduše specifikují základní klíč zdroje pro vložený zdroj bez názvu vlastnosti. Názvy vlastností se odvozují z druhé části názvu.

Implicitní lokalizační výrazy tak můžete používat pouze pro ty vlastnosti, které mají atribut [Localizable]. Dále nefungují pro globální zdroje aplikace. Jiný způsob, jak se dají vlastnosti ovládacích prvků vázat na zdroje, představují explicitní lokalizační výrazy. Nabízejí více flexibility, protože můžete na vložené zdroje vázat jakoukoliv vlastnost ovládacího prvku, a protože fungují i s globálními zdroji aplikace. O explicitních lokalizačních výrazech se více dozvíte v příštím oddílu, kde také uvádíme podrobnosti o globálních zdrojích.

Podíváte-li se pozorně na vygenerované soubory zdrojů na obrázcích 17-7 a 17-8, zjistíte, že ještě nejste hotovi. Přestože je ovládací prvek RegularExpressionValidator zařazen do vygenerovaných zdrojů, vlastnost validačního výrazu tam zařazena není, protože není označena atributem [Localizable]. Ověřování platnosti data narození i roční mzdy však musí brát v úvahu nastavení kultury uživatele, který si stránku prohlíží, protože návštěvníci z USA budou chtít zadávat datum svého narození v tom tvaru, na jaký jsou zvyklí (a nejinak tomu bude u návštěvníků z Německa, Rakouska či z jiných zemí).

Takže – abychom lokalizaci naší aplikace dokončili, budeme muset ještě trochu zapracovat. V zásadě máme na výběr dva způsoby, jak lokalizovat ověřování platnosti hodnot těchto dvou textových polí. Prvním způsobem je automaticky vygenerovat regulární výraz pro ověřování platnosti hodnot, které bude založeny na objektu CultureInfo vytvořeného pro kulturu uživatele. Druhý přístup spočívá v tom, že do vložených zdrojů se přidá položka pro validační výraz. Protože chceme v tomto oddílu probrat, jak fungují explicitní lokalizační výrazy, ukážeme si ten druhý přístup.

Nejprve musíte do vložených zdrojů přidat dvě nové položky, které budou obsahovat regulární výraz pro ověření platnosti vstupní hodnoty pro datum narození uživatele a roční mzdy. Pak je potřeba změnit definici

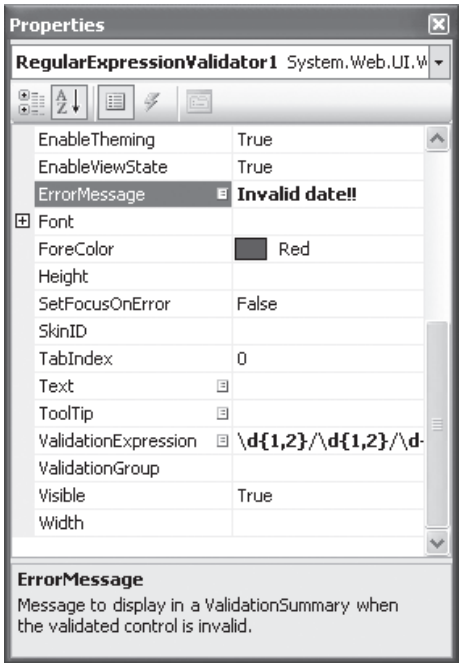
validačních ovládacích prvků – předpokládáme, že se položky zdrojů jmenují jako RegularExpressionValidatorResource1.Validation a RegularExpressionValidatorResource2.Validation:

```
<asp:RegularExpressionValidator
ControlToValidate="BirthdateText" ErrorMessage="Invalid date!!"
ID="RegularExpressionValidator1" runat="server"
ValidationExpression=
'<%$ Resources: RegularExpressionValidatorResource1.Validation %>'
meta:resourcekey="RegularExpressionValidatorResource1" />
```

Patrně jste si už všimli, že předchozí validátor pořád obsahuje nějaký statický text — v tomto případě hodnotu ErrorMessage. S tím si ale žádné starosti nedělejte. Protože validační ovládací prvek má atribut meta:resourcekey, ovládací prvek bude ignorovat statický text, a runtime získá jeho data z vygenerovaných zdrojů. Jakmile má ovládací prvek nějaký takový atribut meta:resourcekey, statický text bude ignorován a všechny informace budou získány z vložených lokalizovaných zdrojů. U ValidationExpression jsme museli použít explicitní lokalizační výrazy, protože automatická lokalizace nebyla pro tuto vlastnost poskytnuta. Obecný formát explicitních lokalizačních výrazů má tuto syntaxi:

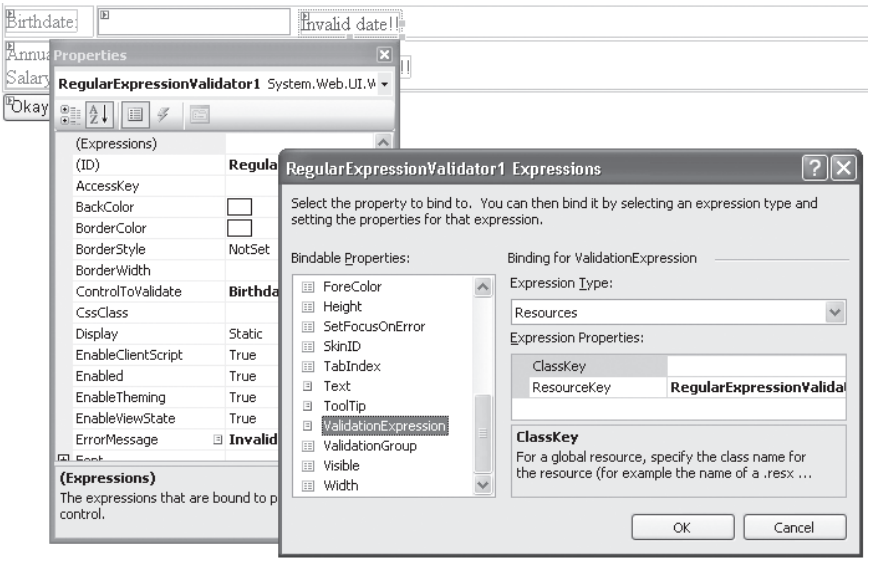
```
<%$ Resources: [KlicAplikace, ] KlicProstredku %>
```

Protože klíč aplikace identifikuje sdílené zdroje aplikace, může se vynechat, když se přistupuje k lokálním zdrojům. Na obrázku 17-11 je vidět, že v okně Properties jsou lokalizované vlastnosti označeny speciální ikonou. Lokalizační výrazy samotné jsou vybaveny novým enginem výrazů dodávaným společně s ASP.NET 2.0.



Obrázek 17-11. *Lokalizované atributy označené v okně Properties.*

U všech vlastností můžete nyní výrazy upravovat v novém editoru výrazů. Okno Properties má nahoře novou vlastnost Expressions. Kliknete-li na tlačítko "tři tečky", otevře se editor výrazu, viz obrázek 17-12. První vlastnost ClassKey specifikuje parametr ApplicationKey pro explicitní lokalizační výraz, druhá vlastnost specifikuje klíč zdroje. Momentálně specifikuje vygenerovaný název třídy globálních zdrojů.



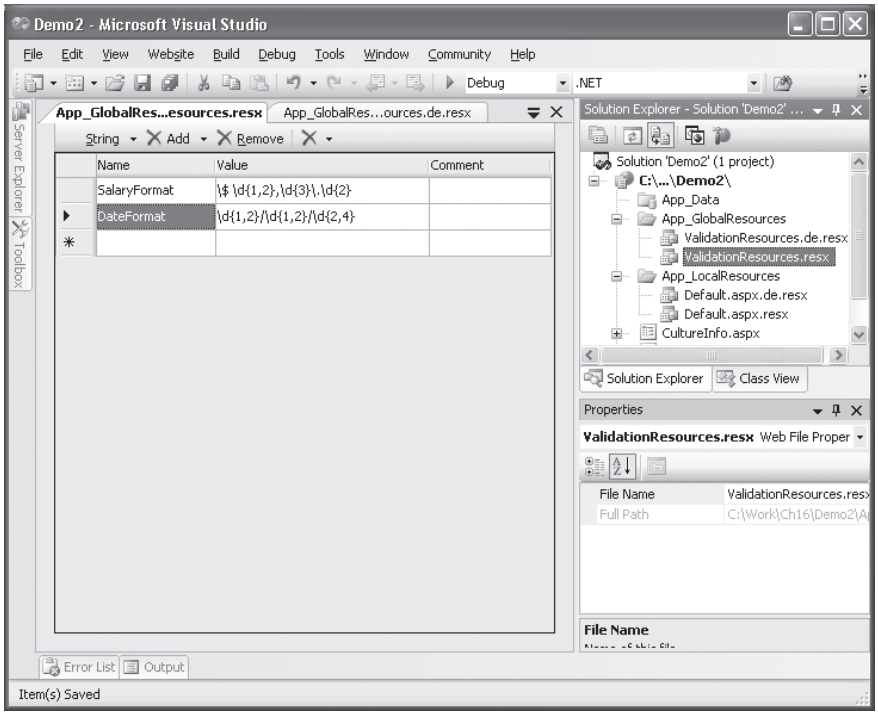
Obrázek 17-12. *Nový editor výrazu.*

Sdílení zdrojů mezi stránkami

Když se generují lokální zdroje pro jednotlivé stránky, mohou vznikat duplicitní řetězce zdrojů nebo jiné duplicity ve zdrojích. Proto se rozhodně vyplatí sdílet zdroje mezi stránkami prostřednictvím globálních zdrojů.

Podpora sdílených zdrojů, pokud jde o nástroje pro generování a vázání zdrojů na ovládací prvky, není tak dobrá jako u lokálních zdrojů stránky. V prvním příkladu této kapitoly jste viděli, jak se používají globální zdroje aplikace, které může sdílet více stránek. Přidáte-li do projektu nějakou assembly zdrojů, Visual Studio se zeptá, zdali má tyto zdroje umístit do adresáře globálních zdrojů. Pokud jsou zdroje v tomto adresáři, jsou dostupné ze všech stránek.

Protože výrazy pro ověřování platnosti hodnot (a také formáty pro datum a pro čísla), které jsme používali dříve, rozhodně patří mezi to, co se bude patrně opětovně využívat ve více stránkách aplikace, bude velmi užitečné, když je dáme do globálního souboru zdrojů. Pro tento účel postačí, když vytvoříme nový globální soubor zdrojů, a k těmto zdrojům pak přidáme hodnoty, jak je to vidět na obrázku 17-13.



Obrázek 17-13. *Globální zdroje přidané do aplikace.*

Nyní musíme zavést explicitní lokalizační výrazy pro naše dva validační ovládací prvky. K tomuto účelu budeme muset změnit název a přidat parametr ApplicationKey. Protože jsme dříve pojmenovali globální soubor zdrojů jako ValidationResources.resx, bude toto hodnota pro vlastnost ApplicationKey (bez přípony .resx).

```
<asp:RegularExpressionValidator ControlToValidate="BirthdateText"
ErrorMessage="Invalid date!!" ID="RegularExpressionValidator1"
runat="server" ValidationExpression='<%$ Resources:ValidationResources,
DateFormat %>' meta:resourcekey="RegularExpressionValidatorResource1" />
```

Dalším velkým rozdílem mezi lokálními a globálními zdroje je to, jak se k nim přistupuje programátorsky. I když se ke globálním zdrojů dá přistupovat se striktními typy prostřednictvím vygenerované třídy zdroje stránky, k lokálním zdrojům se musí přistupovat prostřednictvím metody GetLocalResourceObject.

```
this.GetLocalResourceObject("LabelResource1.Text")
```

POZNÁMKA Lokalizační výrazy a jiné typy výrazů fungují pouze pro ty ovládací prvky, které mají atribut runat="server". Na straně serveru je zpracovává lokalizační infrastruktura ASP.NET. Chcete-li tedy lokalizovat ovládací prvky HTML pomocí implicitních nebo explicitních lokalizačních výrazů, musíte jim přidat atribut runat="server".

Visual Studio v zásadě generuje zdroje pro každou vlastnost s atributem [Localizable] každého ovládacího prvku na stránce nebo uživatelského ovládacího prvku, který je otevřený v režimu návrhu, s už popsaným

formátem názvu Kliczdroje.Nazevvlastnosti. Je-li nějaká vlastnost už svázána s nějakým ovládacím prvkem prostřednictvím explicitního lokalizačního výrazu, generování zdroje pro tuto vlastnost bude kompletně vynecháno. Proto se pro lokalizaci ostatních vlastností uvede v ovládacím prvku atribut meta:resourceKey.

Mimochodem – Visual Studio automaticky vygeneruje položku zdroje pro titulek stránky a přidá do direktivy Page atribut meta:resourceKey. V direktivě Page také můžete aplikovat explicitní lokalizační výrazy, podobně jako zde:

```
<%@ Page Language="C#" ... Culture="auto" UICulture="auto"
meta:resourcekey="PageResource1" %> <%@ Page Language="C#" ... Culture="auto"
UICulture="auto" Title='<%$ Resources:ValidationResources, DateFormat %>' %>
```

První příklad sváže implicitně vlastnosti stránky, jako je titulek, k lokálním zdrojům, druhý příklad sváže vlastnost titulku stránky s globálním prostředkem pojmenovaným jako DateFormat, který je uložen v globální třídě ValidationResources.

A kromě toho, že můžete nastavení pro výchozí kulturu a UI kulturu specifikovat v konfiguračním souboru web.config, jak jsme to předvedli dříve, můžete je také uvést na úrovni stránky.

Lokalizace statického textu

V předchozím oddílu jsme probrali, jak se lokalizují ovládací prvky HTML, když se jim přidá atribut runat="server". Protože se zpracovávají na serveru, lokalizují se vlastnosti velmi snadno. Co ale udělat se statickými texty (a také se směrem textu)?

Odpověď je jednoduchá. ASP.NET 2.0 obsahuje nový ovládací prvek, který je založen na dobře známém ovládacím prvku Literal, který se objevil už v první verzi ASP.NET. Stačí, když statický text zabalíte do tohoto nového ovládacího prvku Localize, jak to předvádí následující fragment kódu:

```
<asp:Localize runat="server"
meta:resourcekey="LiteralResource1"> is some static text!</asp:Localize>
```

Jestliže se tento ovládací prvek obalí okolo nějakého textu, zařadí se taková textová část stránky automaticky do procesu generování zdrojů, jako u všech ostatních ovládacích prvků. Hlavním rozdílem mezi ovládacími prvky Literal a Localize je chování v designérovi Visual Studia. Obsah ovládacího prvku Literal není možné v designérovi editovat, kdežto text zabalený do ovládacího prvku Localize lze editovat stejně jako jakýkoliv jiný textový obsah stránky.

Směr textu

V mezinárodních aplikacích také potřebujete určit směr textu, protože některé kultury čtou text zleva doprava, a jiné zase zprava doleva.

V ASP.NET se to dá zařídit pomocí několika ovládacích prvků, mezi které patří Panel a WebPart. Bude tedy rozumné, když pro směr textu definujete nějakou vlastnost v globálních nebo v lokálních zdrojích, a pro nastavení vlastnosti směru textu ovládacích prvků použijete explicitní lokalizační výrazy. Chcete-li tuto vlastnost nastavit v kořenovém prvku vašeho HTML souboru, musíte přidat atribut runat="server" do samotné značky <html>. Pak na ni můžete aplikovat explicitní lokalizační výrazy, jak to vidíte v následujícím fragmentu kódu:

```
<html runat="server"
      dir='<%$ Resources:ValidationResources,TextDirection %>'
```

```
      xmlns="http://www.w3.org/1999/xhtml" >
...
</html>
```

Shrnutí

V této kapitole jste se dozvěděli základní informace o tom, jak se s ASP.NET vytvářejí mezinárodní webové aplikace. Nejprve jste se dozvěděli, jak .NET spravuje zdroje pro aplikaci, a jak se dá k těmto zdrojům přistupovat programátorsky. Pak jste viděli, že zdroje jsou užitečné nejenom pro lokalizaci, ale také pro řadu jiných věcí, jako jsou vložené výchozí šablony výstupních sestav (což může posloužit jako východisko z nouze, jestliže v nějakých jiných adresářích žádné šablony nenajdete), dodatečné instalační skripty, nebo fragmenty XML, které bude moci využít vaše aplikace.

V druhé části kapitoly jste se dozvěděli, jak CLR spravuje zdroje, které jsou specifické pro dané kultury. CLR vybírá vložené zdroje na základě objektu CultureInfo, který se nastaví vlastností Thread.CurrentUICulture. Obsahuje mechanizmus, který si umí poradit v nouzi (je založený na tzv. paradigmatu hub and spoke), protože podle definované hierarchie kultur hledá zdroje od těch, které jsou nejblíže k aktuální kultuře postupně až k těm, které jsou nejvíce obecné. Nenajde-li aplikace souhlasnou kulturu, při lokalizaci použije výchozí kulturu.

A nakonec jste se dozvěděli, jak ASP.NET a Visual Studio podporují lokalizaci prostřednictvím lokálních zdrojů stránky a sdílených zdrojů aplikace. Dozvěděli jste se také, jak se přistupuje k oběma druhům zdrojů programátorsky i deklarativně prostřednictvím implicitních a explicitních lokalizačních výrazů.