

# KAPITOLA 32

## ASP.NET AJAX

V předchozí kapitole jste vstoupili do světa programování na straně klienta. Naučili jste se několika základním technikám použití JavaScriptu a uvažovali jste nad tím, jak prostřednictvím technik Ajaxu vytvořit rychleji reagující stránky – a to buď sami, nebo prostřednictvím funkcionality zpětného volání klienta v ASP.NET.

Tyto příklady demonstrovaly základy, ze kterých můžete vyjít při vytváření různě pokročilých stránek. Bohužel, programovací model obsahuje ještě spoustu věcí, které je možné zlepšovat. Pokud se výhradně spoléháte na JavaScript, je pouze na vás, abyste přemostili mezeru mezi serverovou abstrakcí ASP.NET a mnohem omezenějším HTML DOM. Není to jednoduché. Bez výhod IntelliSense Visual Studia a jeho ladicích nástrojů, je obtížné napsat bezchybný kód a diagnostikovat problémy. Vytvoření kódu, který bude fungovat ve všech moderních prohlížečích, rovněž představuje výzvu, protože zde rozhodně není nouze o menší výstřednosti a implementační odlišnosti napříč prohlížeči.

Funkcionalita zpětného volání klienta ASP.NET tyto problémy částečně řeší tím, že vám poskytuje serverový model, s jehož pomocí můžete generovat potřebný kód, který je vykonáván na straně klienta (konkrétně se jedná o kód, jenž prostřednictvím objektu `XMLHttpRequest` vykoná asynchronní požadavek). Model zpětného volání klienta má ovšem do dokonalosti daleko. Rozhraní vypadá trochu neobratně, integrace do modelu stránky je trochu nešikovná, přičemž neexistuje ani typování dat. Je pouze na vás, abyste vymysleli způsob, jak serializovat informace, které potřebujete přenášet, do jediného řetězce. Sami také musíte vytvořit kód JavaScriptu, který obdrží zpětné volání, deserializuje řetězec a aktualizuje webovou stránku. Celkově vzato je funkcionalita zpětného volání klienta vynikajícím nástrojem pro vytvoření ovládacích prvků podporujících Ajax, ale už méně zajímavým způsobem, jak navrhovat kompletní webové stránky.

Programátoři ASP.NET mají jinou možnost. Mohou totiž používat sadu nástrojů ASP.NET AJAX, které poskytují několik funkcionalit, jež vám mohou pomoci s vytvářením ajaxových stránek. V této kapitole podrobně prozkoumáte ASP.NET AJAX a naučíte se, jak jej používat k vytvoření nové generace interaktivních dynamických webových stránek.

## Úvod do ASP.NET AJAX

ASP.NET AJAX se skládá ze dvou klíčových částí – z části na straně klienta a z části na straně serveru. Klientská část je sada knihoven JavaScriptu. Tyto knihovny nejsou žádným způsobem spojeny s ASP.NET. V praxi to znamená, že na webových stránkách je mohou používat i programátoři jiných jazyků než ASP.NET. Kli-

entské knihovny toho nenabízí příliš mnoho, alespoň co se týče funkcionality (nejsou zde například žádné předem vytvořené kousky funkcionalit, které byste mohli jednoduše umístit na vaše webové stránky). Místo toho vytvářejí základ, který je nezbytný pro vývoj stránek prostřednictvím ASP.NET AJAX. Tento základ nejenom rozšiřuje jazyk JavaScriptu, aby vyplnil několik jeho mezer (například přidává podporu dědičnosti), ale také poskytuje určitou základní infrastrukturu (například metody pro správu doby života komponent, manipulaci s běžnými datovými typy, provádění reflexe atd.).

Serverová část ASP.NET AJAX funguje na vyšší úrovni. Obsahuje ovládací prvky a komponenty, které používají klientské knihovny JavaScriptu. Webový formulář, který například obsahuje komponentu `DragPanel` (ze sady nástrojů ASP.NET AJAX), poskytuje uživatelům schopnost přetahovat panel v okně prohlížeče. Na pozadí běží vlastní kód JavaScriptu, který používá knihovny ASP.NET AJAX na straně klienta. Komponenta `DragPanel` ovšem veškerý potřebný kód JavaScriptu vytváří sama, díky čemuž jste ušetřeni problémů spojených s psaním kódu.

ASP.NET AJAX je počátkem nového směru ve vývoji ASP.NET. Předtím, než se v této kapitole posunete dále, bude užitečné získat přehled o všech funkcích, které poskytuje, takže zde je jejich stručný přehled:

- **Rozšíření jazyka JavaScript.** Tato rozšíření poněkud přibližují JavaScript k moderním, objektově orientovaným, programovacím jazykům s podporou jmenných prostorů, dědičnosti, rozhraní, výčtů a reflexe.
- **Vzdálené volání metod.** Tato funkcionality vám umožní získat informace ze serveru bez odeslání celé webové stránky zpět na server. Řeší stejný problém jako funkcionality zpětného volání klienta, o níž jste se více dozvěděli v kapitole 31. Oproti ní ovšem umožňuje pracovat se silně typovými metodami místo plnění všech vašich dat do jediného řetězce.
- **Služby ASP.NET.** Tato funkcionality vám umožní zavolat server pro komunikaci s modelem ASP.NET. V současné době můžete pracovat se dvěma službami ASP.NET – službou, která používá informace formulářové autentizace a službou, jež získává data z aktuálního profilu uživatele.
- **Obnovení části stránky.** Nový ovládací prvek `UpdatePanel` poskytuje způsob, jak specifikovat část stránky, kterou bude možné aktualizovat bez toho, aby bylo nutné posílat na server celou stránku. Ze všeho nejlepší je fakt, že pro řízení procesu aktualizace nemusíte psát žádný kód JavaScriptu.
- **Předem vytvořené ovládací prvky.** Populární sada ovládacích prvků ASP.NET AJAX Control Toolkit obsahuje přes 30 ovládacích prvků, které používají rozšířenou funkcionality ASP.NET AJAX pro dosažení skvělých efektů. Najdete zde například ovládací prvky, které provedou sbalení a rozbalení, přidají dynamické animace, nebo jež podporují automatické dokončování a přetahování. Znovu připomínáme, že tyto ovládací prvky zajišťují všechny potřebné nízkourovňové detaily JavaScriptu.

V této kapitole prozkoumáte všechny tyto funkcionality.

### PŘEMĚNA ATLASU NA ASP.NET AJAX

*Pokud pravidelně sledujete dění kolem ASP.NET AJAX, je možné, že jste kdysi pracovali s jednou jeho beta-verzí, která nesla označení Atlas. ASP.NET AJAX nahrazuje Atlas. Nejprve byl vydán jako samostatná komponenta, kterou šlo používat ve spojení s ASP.NET 2.0. Dnes je plně integrovanou součástí platformy ASP.NET 3.5.*

*Ačkoliv ASP.NET AJAX obsahuje celou řadu nejdůležitějších funkcionalit původního Atlasu, několik funkcionalit bylo ponecháno bez povšimnutí. Jednou takovou (a současně nejvíce pozoruhodnou) funkcionalitou byl značkovací standard založený na XML, který byl označován jako klientský skript (client script).*

... pokračování z předchozí stránky.

Tento klientský skript poskytoval deklarativní způsob, jak ve stránce definovat ovládací prvky, které mohou být manipulovány prostřednictvím klientského kódu (podobně jako značky ovládacích prvků ASP.NET definují serverové objekty, s nimiž lze manipulovat prostřednictvím serverového kódu). Pokud jste například vytvořili stránku se dvěma serverovými ovládacími prvky `TextBox` a chtěli klientovi zpřístupnit text v těchto dvou textových polích, nadefinovali byste je v klientském bloku skriptu. Klientský blok skriptu dále poskytoval cestu k dalším funkcionalitám Atlasu, např. chování (což jsou deklarativní funkce jako automatické dokončování či zpracování událostí myši), vázání dat na straně klienta nebo animace. Další informace o Atlasu můžete najít v předchozím vydání této knihy (ASP.NET 2.0 a C# - tvorba dynamických stránek profesionálně, Zoner Press).

Když se Atlas rozvinul do ASP.NET AJAX, opustil standard klientského skriptu společně se souvisejícími funkcionalitami (např. vázání dat), což je jistě škoda. Některé z těchto funkcionalit jsou ovšem dostupné ve vydání ASP.NET AJAX Futures. (Verze Futures poskytuje rozpracované technologie, které mohou být v budoucnu eventuálně integrovány do jádra platformy .NET.) Existují ovšem významné pochybnosti o tom, zdali bude klientský skript v budoucnosti znovu uveden. Přináší totiž nejenom nové komplikace, ale také se překrývá s jiným XML standardem – výkonnějším jazykem XAML, který používají aplikace Silverlightu (viz kapitola 33).

## ASP.NET AJAX na klientovi – knihovny skriptů

Klientská část ASP.NET AJAX se spoléhá na malou kolekci souborů JavaScriptu. Existují dva způsoby, jak rozmísťovat soubory skriptu ASP.NET AJAX. Pokud vytváříte nějakou aplikaci ASP.NET 3.5, jsou vždy umístěny v assembly `System.Web.Extensions.dll` a provedou se na požádání. Pokud nevytváříte aplikace v ASP.NET nebo pokud přidáváte funkcionalitu na straně klienta do běžné stránky HTML, můžete si stáhnout soubory JavaScriptu samostatně z webu ASP.NET AJAX ([http://ajax.asp.net/downloads/default.aspx](http://ajax.asp.net/downloads/defaul.aspx)) jako část knihovny Microsoft AJAX Library.

Pokud si stáhnete knihovnu Microsoft AJAX Library, zjistíte, že ASP.NET AJAX používá pouze tři základní soubory JavaScriptu – `MicrosoftAjax.js`, `MicrosoftAjaxWebForms.js` a `MicrosoftAjaxTimer.js`. Společně s těmito základními soubory se zde nachází více než 100 velmi malých souborů JavaScriptu, ve kterých jsou uloženy globalizační informace (například datové formáty).

---

**TIP** Microsoft AJAX Library stojí za stažení i tehdy, pokud se chcete blíže podívat na skutečný kód JavaScriptu. Najdete zde nejenom ladící verzi každého ze tří základních souborů, ale také jejich finální verze. Tyto produkční verze mají odstraněna všechna prázdná místa a komentáře, aby výsledné soubory mohly být co nejmenší. Pro porovnání: největším souborem je `Microsoft.Ajax.js`, jehož ladící verze má 254 KB. Produkční verze má pouze 82 KB.

---

V ASP.NET nenaleznete jednotlivé soubory JavaScriptu pro klientské knihovny. Klientské knihovny jsou umístěny přímo v assembly `System.Web.Extensions.dll`, přičemž jsou poskytovány jako skriptový zdroj. Skriptový zdroj je podobný webovým zdrojům, o nichž jste se dozvěděli v kapitole 28. Skriptové zdroje vám (podobně jako webové zdroje) umožní mapovat URL na zdroj, který je umístěn v assembly. Zde je například ukázka bloku skriptu, který rozbálí knihovnu skriptů ASP.NET AJAX.

```
<script src="/YourWebSite/ScriptResource.axd?d=RUSU1mIv69CJ9H5JUA0Sw8L4674
LfxG0Qg6Nw7HtNHheB3bMiw70v16bX1KPG6N1oTYEi65ggRoIP1hWapSttV3udoNXGrk095YGEzuX0M1&am
p;t=633127440334523405" type="text/javascript">
</script>
```

Pokud se podíváte do souboru `web.config` na nějakou aplikaci ASP.NET 3.5, určitě najdete mapování, které linkuje požadavek na `ScriptResource.axd` na třídu `System.Web.Handlers.ScriptResourceHandler`, jež je uložena v assembly `System.Web.Extensions.dll`.

```
<handlers>
...
<add name="ScriptResource" preCondition="integratedMode" verb="GET,HEAD"
    path="ScriptResource.axd"
    type="System.Web.Handlers.ScriptResourceHandler ..." />
</handlers>
```

Třída `ScriptResourceHandler` prozkoumá předaný argument dotazovacího řetězce a vrátí požadovaný soubor skriptu. Stručně řečeno – `ScriptResourceHandler` zpracuje skriptové zdroje stejným způsobem, jakým `WebResourceHandler` zpracuje webové zdroje.

## ASP.NET AJAX na serveru – ScriptManager

S velkou pravděpodobností nebudete chtít na každé stránce, která vyžaduje komponentu ASP.NET AJAX, ručně vypisovat dlouhé adresy URL, jež ukazují na skriptové zdroje. V takovém případě je dobrým řešením použít ovládací prvek `ScriptManager` ASP.NET.

`ScriptManager` je mozkem serverového modelu ASP.NET AJAX. Jedná se o webový ovládací prvek, který nemá vůbec žádnou vizuální podobu na stránce. Provádí však klíčovou úlohu – realizuje odkazy na javascriptové knihovny ASP.NET AJAX.

### SKRIPTOVÉ ZDROJE VERSUS WEBOVÉ ZDROJE

*Všechno tohle vyvolává jednu vynikající otázku – proč ASP.NET, které obsahuje systém pro webové zdroje, zahrnuje i podobný systém pro skriptové zdroje? Soubory JavaScriptu je totiž možné vložit do assembly prostřednictvím běžných webových zdrojů.*

*Odpověď je tato – skriptové zdroje přidávají určitá zdokonalení. Pokud je požadujícím webovým prohlížečem IE 7, skriptové zdroje automaticky použijí komprimaci pro rychlejší stahování. Kromě toho se `ScriptResourceHandler` trochu více elegantněji zapojí do infrastruktury ASP.NET AJAX na straně klienta. Jakmile se soubor skriptu načte, `ScriptResourceHandler` zavolá funkci `Sys.Application.notifyScriptLoaded()` na straně klienta. Pokud používáte soubor skriptu, který se poskytuje prostřednictvím webového zdroje, nebo soubor skriptu, jenž vůbec není vnořen do assembly, je pouze na vás, abyste na konci vašeho skriptu zavolali funkci `Sys.Application.notifyScriptLoaded()`. Tento krok oznámí pracovnímu rámci ASP.NET AJAX na straně klienta, že skript byl načten, takže bude možné zpracovávat další skript. V případě, kdy tuto akci neprovedete, nebudou některé rysy ASP.NET AJAX v určitých prohlížečích správně fungovat.*

Pokud chcete přidat na vaši stránku prvek `ScriptManager`, můžete jej přetáhnout ze záložky `AJAX Extensions toolboxu` Visual Studia. Podívejte se na deklaraci prvku `ScriptManager` v souboru `.aspx`:

```
<asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
```

Každá stránka, která využívá funkcionality ASP.NET AJAX, vyžaduje instanci prvku `ScriptManager`. Jedna stránka může obsahovat pouze jeden prvek `ScriptManager`. Vedle realizace odkazů na klientské knihovny ASP.NET AJAX má `ScriptManager` na starosti i některé další důležité úlohy. Může realizovat odkazy na jiné

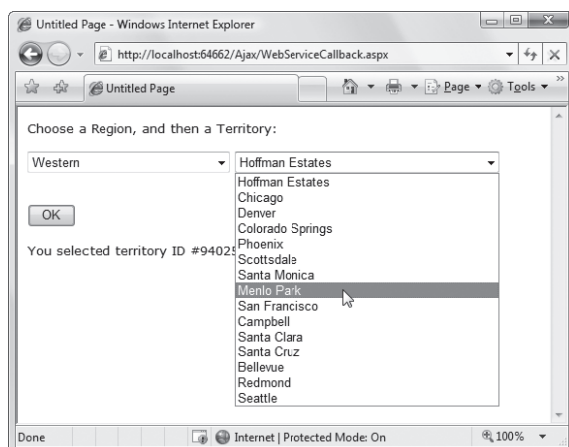
soubory skriptu (často na požadavek jiných ovládacích prvků a komponent ASP.NET AJAX), vytvářet proxy, které vám umožní asynchronně volat webové služby z prohlížeče, nebo řídit způsob, jakým bude prvek UpdatePanel obnovovat obsah. V této kapitole prozkoumáme všechna tato témata.

**TIP** Pokud na vašem webu využíváte nějakou funkcionalitu ASP.NET AJAX, je docela rozumné umístit prvek ScriptManager do vzorové stránky (master page). To ovšem může někdy působit problémy, protože rozdílný obsah stránek může vyžadovat odlišnou konfiguraci vlastností prvku ScriptManager (například přidání nových skriptů nebo nových odkazů na webové služby). V takové situaci řešení spočívá v použití prvku ScriptManager na vzorové stránce a prvku ScriptManagerProxy na obsahové stránce. Na každé takové obsahové stránce můžete konfigurovat ovládací prvek ScriptManagerProxy stejným způsobem jako prvek ScriptManager.

Nyní, když jsme vám sdělili základní informace o modelu ASP.NET AJAX (tedy o klientských knihovnách a serverovém ovládacím prvku ScriptManager), jste připraveni začít vytvářet stránky, které budou využívat různé funkcionality ASP.NET AJAX. Jako úplně první věc vám předvedeme, jak využít ASP.NET AJAX pro vytvoření alternativy k technice zpětného volání klienta za účelem získání informací ze serveru. Dále vám ukážeme, jak se dají webové ovládací prvky s podporou ASP.NET AJAX využít pro získání typických ajaxových efektů (a to s malým úsilím). A nakonec se blíže podíváme na klientské knihovny ASP.NET AJAX, které podporují tyto funkcionality, přičemž vás naučíme, jak vytvořit vlastní komponentu ASP.NET AJAX.

## Serverová zpětná volání

Prvním příkladem použití ASP.NET AJAX, o němž budeme přemýšlet, je revidovaná verze stránky zpětného volání klienta z kapitoly 31. Tato stránka obsahuje dvě pole se seznamy (viz obrázek 32-1). V prvním seznamu se zobrazuje seznam regionů. Ve druhém seznamu se pak zobrazují oblasti vybraného regionu. Trik spočívá v tom, že tento druhý seznam se vyplní pokaždé, když uživatel provede nějaký výběr v prvním seznamu. Vyplnění druhého seznamu se neobejde bez zavolání serveru, který prohledá databázi a poskytne výsledky.



Obrázek 32-1. Příklad dynamického seznamu.

Aby tato stránka mohla využívat funkcionalitu zpětného volání klienta ASP.NET, musíte implementovat trochu těžkopádné rozhraní ICallbackEventHandler. ASP.NET AJAX ovšem používá poněkud jiný pří-

stup. V ASP.NET AJAX jsou zpětná volání vždy vykonávána prostřednictvím samostatné serverové metody – z technického pohledu se jedná o webovou službu. Tento návrh zlepšuje oddělení logiky, což vám pomáhá lépe uspořádat váš kód. Ovšem mnohem důležitější je fakt, že je zajištěn proces serializace. To znamená, že nemusíte vymýšlet vlastní metodu pro zaslání komplexních dat (viz například náš neohrabaný systém pro rozdělení jednotlivých hodnot řetězce, který byl popisován v kapitole 31).

V následujících sekcích vám předvedeme, jak vytvořit webovou službu, kterou potřebujete, přičemž vám rovněž nastíníme několik možností jejího využití.

## Webové služby v ASP.NET AJAX

Při vykonávání serverového zpětného volání pomocí ASP.NET AJAX volá váš javascriptový kód na straně klienta nějakou metodu v serverové webové službě.

Webová služba je kolekcí jedné nebo více serverových metod, které mohou být volány vzdálenými klienty. Pro zavolání webové služby klient zasílá zprávu s požadavkem prostřednictvím HTTP (podobně jako v procesu odeslání webové stránky, ovšem s tím rozdílem, že tělo požadavku obsahuje argumenty, které se předávají metodě). ASP.NET poté vytvoří objekt webové služby, který spustí kód v odpovídající webové metodě, vrátí výsledek. Objekt webové služby bude poté zničen. Formát zprávy pro požadavek a odpověď bývá různý. Tradičně se jedná o XML standard označovaný jako SOAP, ovšem v ASP.NET AJAX se jedná o odlehčenou textovou alternativu označovanou jako JSON (JavaScript Object Notation), která se používá hlavně z důvodu lepší kompatibility mezi prohlížeči.

---

**TIP**      *Další informace o webových službách můžete najít na webu vydavatelství Zoner Press (<http://www.zonepress.com>). V sekci Download můžete najít tři bonusové kapitoly (ve formátu PDF), které podrobněji popisují protokoly webové služby, architekturu webové služby a postup, jak vytvořit a používat webové služby ASP.NET. Tyto kapitoly jsou v češtině a pocházejí z předchozího vydání této knihy.*

---

Je důležité si uvědomit, že ačkoliv mechanismus zpětného volání ASP.NET AJAX používá webové služby, specializuje se hlavně na implementaci. Pokud jste trochu obeznámeni s webovými službami, nepochybně zjistíte, že ASP.NET AJAX zavádí určité speciální omezení. První omezení – webová stránka nemůže volat jiné webové služby než webové služby ASP.NET AJAX (například webové služby třetích stran, které byly vytvořeny na jiných platformách). Důvodem je skutečnost, že tyto jiné webové služby nepodporují zjednodušený model JSON, který se používá v ASP.NET AJAX. Druhé omezení – webová stránka nemůže volat webové služby, které jsou umístěny v jiných doménách (jinak řečeno – na jiných webových serverech). Je to kvůli tomu, že většina webových prohlížečů nedovoluje použití objektu XMLHttpRequest napříč různými doménami (čímž se předchází některým potenciálním skriptovacím útokům).

Tato omezení pochopitelně vůbec neomezují původně zamýšlené použití funkcionality zpětného volání (ve smyslu mechanismu stránky k provádění aplikačních úloh na straně serveru). Pokud ovšem pomocí webových služeb plánujete vystavovat serverovou funkcionalitu klientům, vývojářům třetích stran či aplikacím, které nebudou na .NET, musíte si uvědomit, že toto vůbec není cílem webových služeb v ASP.NET AJAX.

---

**POZNÁMKA**      *Samozřejmě existují určité způsoby, jak tato omezení obejít. Ve vaší webové aplikaci můžete například zavolat nějakou webovou metodu, která následně zavolá nějakou jinou webovou metodu, jež existuje v jiné doméně. Tato přemostující technika funguje, protože kód webového serveru nemá stejná omezení jako prohlížeč na straně klienta. Jinak řečeno – ze serveru je možné libovolně volat různé webové služby v různých doménách.*

---

## Vytvoření webové služby

Ačkoliv se webové služby na stránkách ASP.NET AJAX používají speciálním způsobem, jsou definovány stejným způsobem. Stejně jako všechny ostatní webové služby ASP.NET i webové služby, které budete používat s ASP.NET AJAX, se skládají ze dvou částí: souboru `.asmx`, který vystupuje jako koncový bod webové služby a souboru `.cs`, jenž obsahuje skutečný kód C#. Tyto soubory musíte vložit na web, který obsahuje stránku ASP.NET AJAX, jež bude používat danou webovou službu.

Nejrychlejším způsobem, jak vytvořit webovou službu ve Visual Studiu, je zvolit Website -> Add New Item (nebo Project -> Add New Item for web projects), vybrat šablonu Web Service, zadat název souboru (v následujícím příkladě se jedná o `TerritoriesService`) a klepnout na Add. Pokud vytváříte web bez projektu, soubor `.asmx` bude umístěn v adresáři webové aplikace, zatímco odpovídající soubor `.cs` bude umístěn do složky `App_Code`, aby mohl být automaticky zkompilován.

---

**POZNÁMKA** *K tomu, abyste mohli použít webovou službu s ASP.NET AJAX, nemusíte webovou aplikaci hostovat ve virtuálním adresáři IIS. Místo toho ji můžete otestovat pomocí integrovaného webového serveru ve Visual Studiu. Tento postup funguje, protože kód skriptu, který volá webovou službu, automaticky používá relativní cestu. V důsledku toho bude stránka schopna sestavit správnou URL, bez ohledu na to, jaký port zvolí webový server Visual studia.*

---

Na souboru `.asmx` není nic mimořádného – pokud jej otevřete, najdete zde jediný řádek s direktivou `WebService`, která specifikuje jazyk kódu, umístění souboru s kódem v pozadí a název třídy:

```
<%@ WebService Language="C#" CodeBehind="~/App_Code/TerritoriesService.cs"
    Class="TerritoriesService" %>
```

V tomto příkladu je vytvořena webová služba s názvem `TerritoriesService.asmx` se souborem kódu v pozadí `TerritoriesService.cs`. Třída kódu v pozadí definuje třídu s názvem `TerritoriesService`, která vypadá následovně:

```
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.Web.Script.Services.ScriptService]
public class TerritoriesService : System.Web.Services.WebService
{ ... }
```

Tato třída je odvozena ze `System.Web.Services.WebService`, což je tradiční základní třída pro webové služby. Jedná se ovšem pouze o konvenci, která není nutná. Odvozením ze třídy `WebService` získáte přístup k některým vestavěným objektům (např. `Application`, `Server`, `Session` a `User`) bez potřeby procházet statickou vlastnost `HttpContext.Current`.

Povšimněte si skutečnosti, že deklarace třídy webové služby obsahuje tři atributy. První dva atributy – `WebService` (nastavuje jmenný prostor XML, který se používá ve zprávách webové služby) a `WebServiceBinding` (indikuje úroveň shody se standardy, jež webová služba podporuje) – se použijí pouze tehdy, pokud voláte webovou službu pomocí zpráv SOAP. Tyto atributy se nevztahují na stránky ASP.NET AJAX. Třetí atribut – `ScriptService` – je ovšem mnohem důležitější. Konfiguruje totiž webovou službu tak, aby povolila volání JSON z klientů JavaScriptu. Bez tohoto detailu nebude možné na stránce ASP.NET AJAX používat webovou službu.



**POZNÁMKA**      Ve výchozím nastavení je atribut `ScriptService` okomentován. Abyste mohli vytvořit webovou službu, kterou lze volat ze stránky ASP.NET AJAX, ujistěte se, že jste odstranili značky pro komentáře.

## Vytvoření webové metody

Nyní jste připraveni napsat kód pro vaši webovou službu. Každá webová služba obsahuje jednu nebo více metod, které jsou označeny atributem `WebMethod`. Atribut `WebMethod` činí metodu vzdáleně volatelnou. Pokud přidáte nějakou metodu, která neobsahuje atribut pro webovou metodu, kód na straně serveru ji sice bude schopen používat, nicméně JavaScript na straně klienta ji nebude schopen přímo volat.

```
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.Web.Script.Services.ScriptService]
public class TerritoriesService : System.Web.Services.WebService
{
    WebMethod()
    public void DoSomething()
    { ... }
}
```

Není nutné vytvářet metodu jako veřejnou (jak je tomu zde). Toto se obvykle dělá kvůli konvenci a jasnosti.

Webové metody mají určitá omezení. Datové typy, které používáte pro hodnoty parametrů a návratové hodnoty, musí používat jeden z datových typů, jež jsou uvedeny v tabulce 32-1.

**Tabulka 32-1. Datové typy webové služby pro parametry a návratové hodnoty.**

Datový typ	Popis
Základní typy	Základní datové typy C#, např. celá čísla ( <code>short</code> , <code>int</code> , <code>long</code> ), nepodepsaná celá čísla ( <code>ushort</code> , <code>uint</code> , <code>ulong</code> ), neintegrální numerické typy ( <code>float</code> , <code>double</code> , <code>decimal</code> ) a několik jiných různých typů ( <code>bool</code> , <code>string</code> , <code>char</code> , <code>byte</code> , <code>DateTime</code> ).
Výčty	Jsou podporovány výčtové typy (definované v C# klíčovým slovem <code>enum</code> ). Webová služba však používá řetězcový název hodnoty výčtu (nikoli podkladové celé číslo).
Vlastní objekty	Můžete předat jakýkoliv objekt, který vytvoříte na základě vlastní třídy nebo struktury. Jediným omezením je to, že se přenáší pouze data veřejných členů a vlastností, přičemž všichni veřejní členové a vlastnosti musí používat jeden z podporovaných datových typů. Pokud používáte třídu, která obsahuje nějaké vlastní metody, tyto metody se na klienta nepřenesou, takže pro něj nebudou dostupné.
Pole nebo kolekce	Můžete používat pole jakéhokoliv podporovaného typu. Můžete také používat <code>ArrayList</code> , který se jednoduše převede na pole. Nemůžete používat specializovanější kolekce jako <code>Hashtable</code> . Můžete používat obecné kolekce. Ve všech těchto případech musí být objekty v kolekci serializovatelné.
XmlNode	Objekty založené na <code>System.Xml.XmlNode</code> jsou reprezentacemi částí dokumentu XML. Tuto skutečnost můžete používat k zasílání libovolného XML.



Datový typ	Popis
Sada dat a datová tabulka	Sadu dat ( <i>DataSet</i> ) a datovou tabulku ( <i>DataTable</i> ) můžete používat pro vrácení informací z relační databáze. Nejsou podporovány jiné datové objekty ADO.NET, jako třeba datové sloupce ( <i>DataColumns</i> ) nebo datové řádky ( <i>DataRow</i> s). Když použijete sadu dat ( <i>DataSet</i> ) nebo datovou tabulku ( <i>DataTable</i> ), automaticky se převede na XML podobným způsobem jako při použití metod <code>GetXml()</code> nebo <code>WriteXml()</code> .

### STAV RELACE VE WEBOVÉ SLUŽBĚ

Atribut `WebMethod` akceptuje několik parametrů, z nichž většina má na stránkách ASP.NET AJAX poměrně malý význam. Jednou výjimkou je vlastnost `EnableSession`, která má standardně hodnotu `false`. Tato vlastnost realizuje stav relace, který může být přístupný vaší webové službě. Výchozí hodnota `false` je rozumná v tradiční webové službě, která nepoužívá ASP.NET AJAX, protože zde nejsou informace o relaci potřebné, přičemž klient nemusí udržovat cookie relace. Ovšem u webové služby ASP.NET AJAX se volání webové služby vždy provede v kontextu webové stránky ASP.NET, která se vykonává v kontextu aktuálního uživatele webové aplikace. To znamená, že tento uživatel má živou relaci a cookie relace, která se automaticky přenáší společně s voláním webové služby. Podívejte se na příklad, který poskytuje webové metodě přístup k objektu `Session`:

```
[WebMethod(EnableSession = true)]
public void DoSomething()
{
    if (Session["myObject"] != null)
    {
        // (Použije objekt ve stavu relace.)
    }
    else
    {
        // (Vytvoří nový objekt a uloží jej do stavu relace.)
    }
}
```

V našem příkladu se dvěma seznamy to znamená, že webová služba musí poskytnout způsob pro získání oblastí, které spadají pod daný region. Následující kód znázorňuje webovou službu, která obsahuje webovou metodu s názvem `GetTerritoriesInRegion()`, která získává takové oblasti:

```
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.Web.Script.Services.ScriptService]
public class TerritoriesService : System.Web.Services.WebService
{
    [WebMethod()]
    public List<Territory> GetTerritoriesInRegion(int regionID)
    {
        SqlConnection con = new SqlConnection(
            WebConfigurationManager.ConnectionStrings[
```

```

        "Northwind"].ConnectionString);
SqlCommand cmd = new SqlCommand(
    "SELECT * FROM Territories WHERE RegionID=@RegionID", con);
cmd.Parameters.Add(new SqlParameter("@RegionID", SqlDbType.Int, 4));
cmd.Parameters["@RegionID"].Value = regionID;
List<Territory> territories = new List<Territory>();
try
{
    con.Open();
    SqlDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        territories.Add(new Territory(
            reader["TerritoryID"].ToString(),
            reader["TerritoryDescription"].ToString()));
    }
    reader.Close();
}
catch (SqlException err)
{
    // Maskuje chyby.
    throw new ApplicationException("Data error.");
}
finally
{
    con.Close();
}
return territories;
}
}

```

Kód v metodě `GetTerritoriesInRegion()` je podobný kódu, který jsme použili v kapitole 31 pro poskytnutí zpětného volání klienta. Zásadním rozdílem tohoto kódu je to, že nevrací jeden dlouhý řetězec s výsledky. Informace jsou vráceny prostřednictvím silně typovaného seznamu objektů `Territory`. To je mnohem lepší způsob, který umožňuje předcházet různým nahodilým chybám.

Třída `Territory` zaobaluje dva kousky řetězcové informace. Místo vlastností používá veřejné členské proměnné, protože funguje výhradně jako datový balík, který přenáší informace po síti:

```

public class Territory
{
    public string ID;
    public string Description;
    public Territory(string id, string description)
    {
        this.ID = id;
        this.Description = description;
    }
}

```

```
}  
public Territory() { }  
}
```

Tuto definici třídy můžete umístit buď do stejného souboru s kódem jako webovou službu, nebo do samostatného souboru v adresáři App\_Code.

## Volání webové služby

Když jste nyní vytvořili webovou službu, kterou potřebujete, musíte ještě nakonfigurovat stránku tak, aby věděla o službě TerritoriesService. K tomuto účelu potřebujete do stránky přidat ovládací prvek ScriptManager. Poté přidejte sekci <Services> do značky pro ovládací prvek ScriptManager. V této sekci jsou pomocí prvků ServiceReference uvedeny všechny služby, které vaše stránka používá a jejich umístění. Podívejte se, jak přidat referenci na soubor TerritoriesService.asmx uvedený dříve:

```
<asp:ScriptManager ID="ScriptManager1" runat="server">  
  <Services>  
    <asp:ServiceReference Path="~/TerritoriesService.asmx" />  
  </Services>  
</asp:ScriptManager>
```

ScriptManager vygeneruje proxy JavaScriptu, kterou můžete použít pro vytvoření vašeho volání. V aktuálním příkladu použijeme tuto proxy pro volání webových metod webové služby TerritoriesService. Zde je řádek kódu JavaScriptu, který volá metodu GetTerritoriesInRegion():

```
TerritoriesService.GetTerritoriesInRegion(regionID, OnRequestComplete);
```

Pokud jste už někdy předtím naprogramovali webovou službu v ASP.NET, povšimnete si, že syntaxe na straně klienta pro volání webové služby v ASP.NET AJAX se trochu liší od syntaxe .NET. V aplikaci .NET musíte nejprve vytvořit objekt proxy a poté na tomto objektu zavolat webovou službu. Na stránce ASP.NET AJAX použijete připravený objekt proxy, který bude mít stejný název jako má třída webové služby.

Volání webové služby na straně klienta jsou asynchronní, takže parametry původní webové metody musíte vždy poskytnout společně s dalším parametrem, který specifikuje funkci JavaScriptu na straně klienta, jež by se měla zavolat při přijetí výsledku. Nepovinně můžete přidat jiné reference na funkce, které se použijí, když je volání dokončeno:

```
TerritoriesService.GetTerritoriesInRegion(regionID, OnRequestComplete, OnError);
```

Posledním krokem, který je zapotřebí vykonat pro dokončení našeho příkladu se seznamy, je přidat kód JavaScriptu, jenž bude volat webovou službu a zpracovávat výsledek. V tomto případě potřebujeme alespoň dvě funkce – jednu pro spuštění zpětného volání a druhou pro přijetí výsledku. Podívejte se na funkci JavaScriptu, která spustí proces:

```
function GetTerritories(regionID)  
{  
  TerritoriesService.GetTerritoriesInRegion(regionID,  
    OnRequestComplete, OnError);  
}
```

Jakákoliv změna výběru v prvním poli se seznamem spustí funkci JavaScriptu, která vykoná zpětné volání a předá hodnotu `regionID` z aktuálního výběru:

```
<asp:DropDownList ID="lstRegions" Runat="server" ...
                    onchange="GetTerritories(this.value);" />
```

Z technického pohledu můžete umístit veškerý kód přímo do funkce `GetTerritories()` v atributu události `onchange` a tím zmenšit počet funkcí JavaScriptu, které musíte napsat. Nicméně oddělením kódu, který volá webovou službu, se nejenom rapidně zlepší jeho čitelnost, ale také udržovatelnost.

Funkce `OnRequestComplete()` se spustí po přijetí odpovědi. Akceptuje návratovou hodnotu prostřednictvím jediného parametru. Poté předá informace do druhého pole se seznamem na webové stránce:

```
function OnRequestComplete(result)
{
    var lstTerritories = document.getElementById("lstTerritories");
    lstTerritories.innerHTML = "";
    for (var n = 0; n < result.length; n++)
    {
        var option = document.createElement("option");
        option.value = result[n].ID;
        option.innerHTML = result[n].Description;
        lstTerritories.appendChild(option);
    }
}
```

Pozoruhodným rysem tohoto kódu je to, že je schopen pracovat s výsledkem, který byl vrácen webovou metodou, bez dalších deserializačních prací. Tohle všechno je ještě působivější, uvažíme-li, že webová metoda vrací obecný seznam objektů `Territory`, což je věc, která nemá odpovídající ekvivalent v kódu JavaScriptu. ASP.NET AJAX místo toho vytvoří definici objektu `Territory`, přičemž v poli vrací celý seznam. To umožní vašemu kódu JavaScriptu procházet pole a prověřovat vlastnosti `ID` a `Description` každé položky.

Existuje jedna menší vychytávka, kterou nyní můžete uplatnit. Místo použití metody `document.getElementById()` můžete použít alias ASP.NET AJAX `$get`, který vykoná stejnou funkci a vypadá následovně:

```
var lstTerritories = $get("lstTerritories");
```

Na webových stránkách ASP.NET AJAX se toto zcela běžné používá.

Tento příklad nyní funguje stejně jako příklad se zpětným voláním z kapitoly 31, ovšem s tím rozdílem, že tato verze používá silně typovanou webovou metodu bez komplikovaného kódu pro serializaci řetězce. Rovněž nemusíte přidávat žádný serverový kód pro získání reference zpětného volání a jeho dynamického vložení.

Můžete používat přímočaré proxy, které poskytují přístup k vaší službě. Jako poslední úpravu přidejte časový limit a funkci pro ošetření chyb, viz níže:

```
function OnError(result)
{
    var lbl = document.getElementById("lblInfo");
    lbl.innerHTML = "<b>" + result.get_message() + "</b>";
}
```