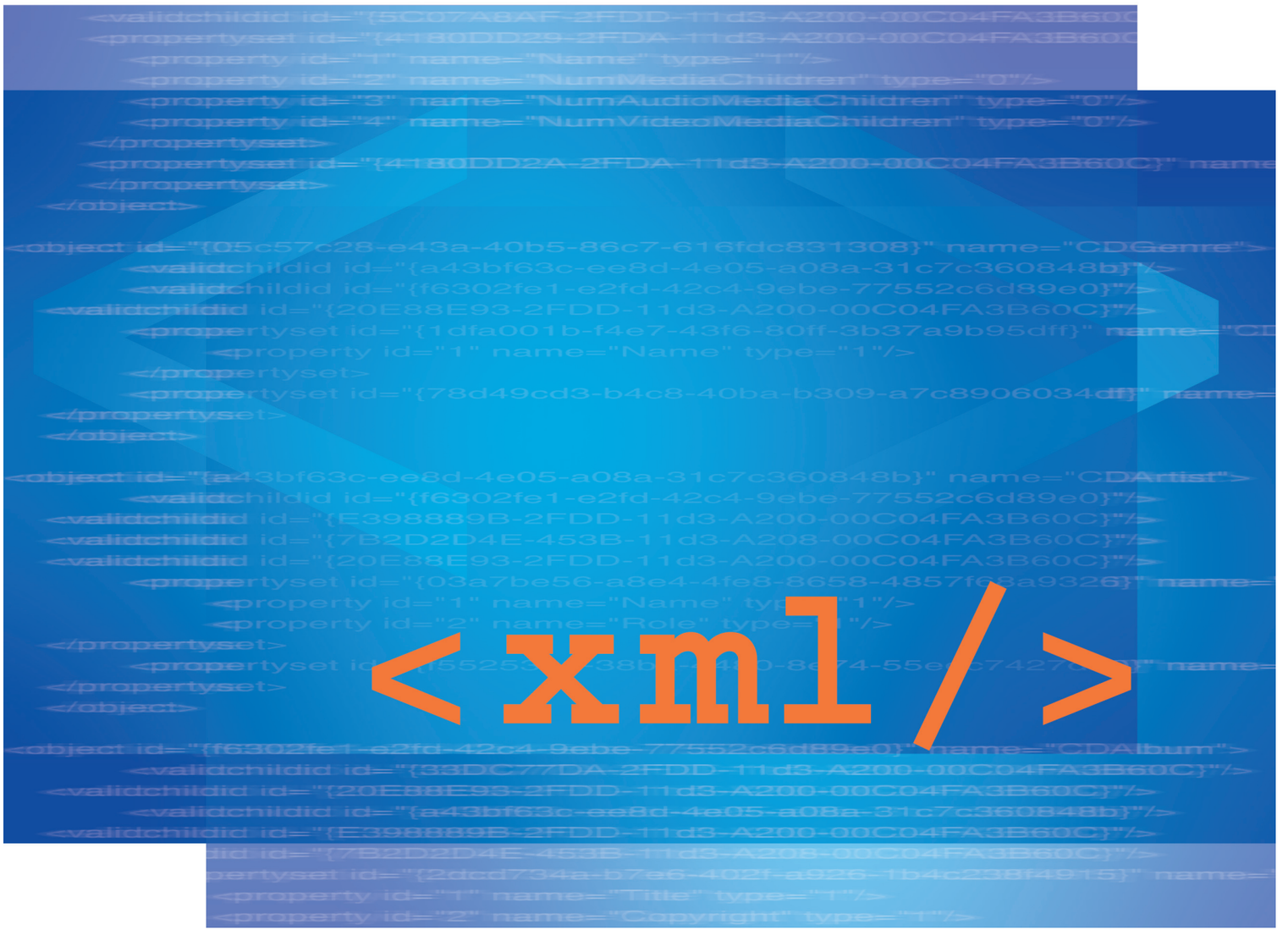
Microsoft .net



Imrich Buranský

XML a webové služby

Prienik do XML cez Microsoft .NET a Murphyho zákony

Microsoft®

XML a webové služby

Materiály a zdrojové súbory cvičných príkladov sú k dispozícii na adrese
http://msdn.microsoft.cz/docs/BrozuraXML.zip

Úvod

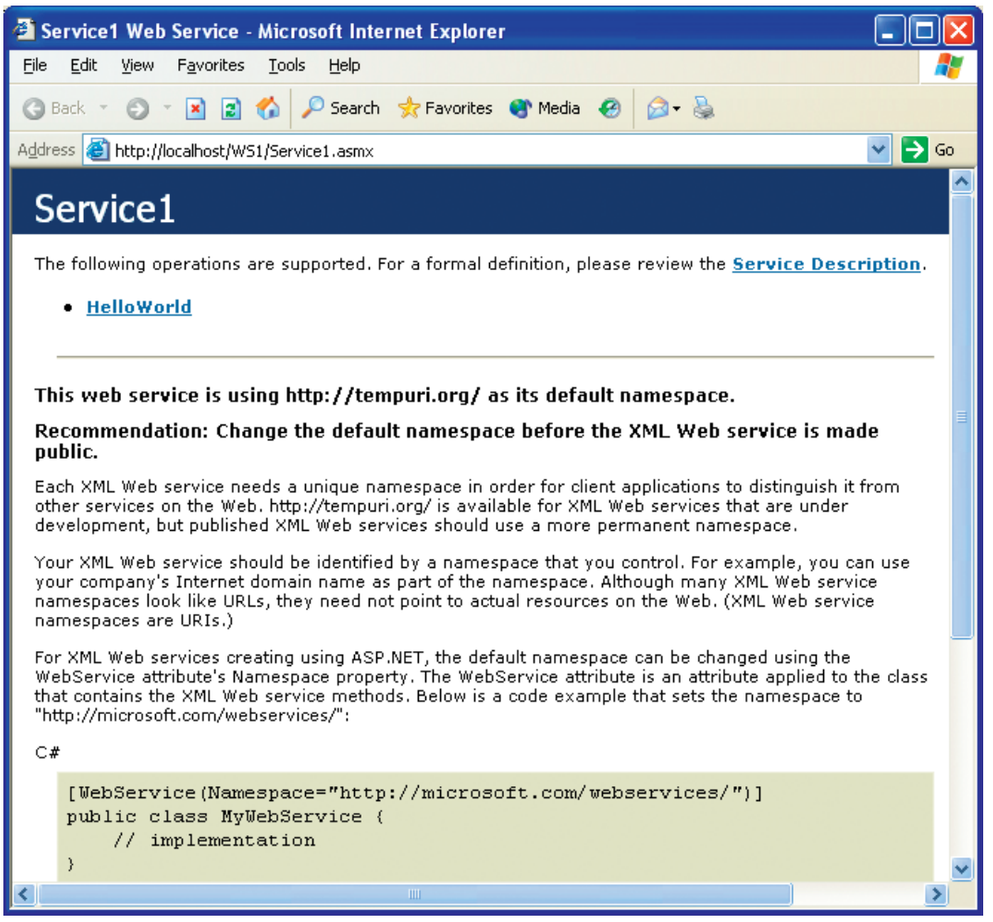
Motto: MURPHYHO ZÁKON
Každé nové riešenie rodí nové problémy.

Neschopnosť dorozumieť sa - mnohojazyčnosť bola príčinou neúspechu dostavby Babylonskej veže. Popletenie jazykov nastalo, keď veža začala dosahovať nebesá. Neprajníci tvrdia, že v prípade počítačov sa mnohojazyčnosť prejavila už pri vymeriavaní základov. Aj napriek tomu sme svedkami nebývalého rozmachu počítačov. V súčasnosti už nejde len o to, aby sa dokázali dohovoriť tvorcovia počítačov, ale aby sa dokázali dorozumieť aj samotné počítače. Pri tom sa veľké očakávania spájajú s webovými službami a s XML - Extensible Markup Language, na ktorom sú vybudované.

Vo vývojovom prostredí firmy Microsoft - Visual Studio .NET je vytvorenie programu, ktorý predstavuje webovú službu, veľmi jednoduché. Stačí zvoliť nový projekt, napr. typ projektu Visual C# Projects, šablónu ASP .NET Web Service, určiť umiestnenie a názov projektu (napr. http://localhost/WS1, kde WS1 je názov projektu). Potom stačí zrušiť komentár vo vygenerovanom zdrojovom kóde, aby sme získali metódu webovej služby:

```
[WebMethod]
public string HelloWorld()
{
  return "Hello World";
}
```

Vybudujme projekt a otestujme ho stlačením F5. V okne Internet Explolera získame údaje, ktoré zachytáva obr. 0.1.



Obr. 0.1. Prvá webová služba.

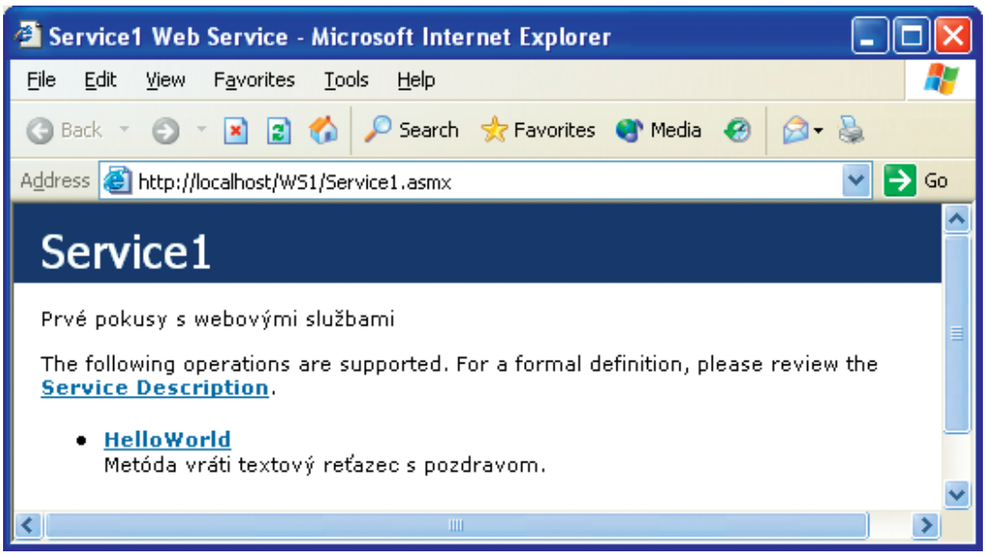
Podľa odporúčania z textu obr. 0.1. definujme k našej webovej službe priestor mien, napr.:

```
[WebService(Namespace="http://buransky.sk/ws1/",
   Description="Prvé pokusy s webovými službami")]
public class Service1 : System.Web.Services.WebService
{
  // Tu je implementácia ...
}
```

Doplňme tiež opis metódy webovej služby:

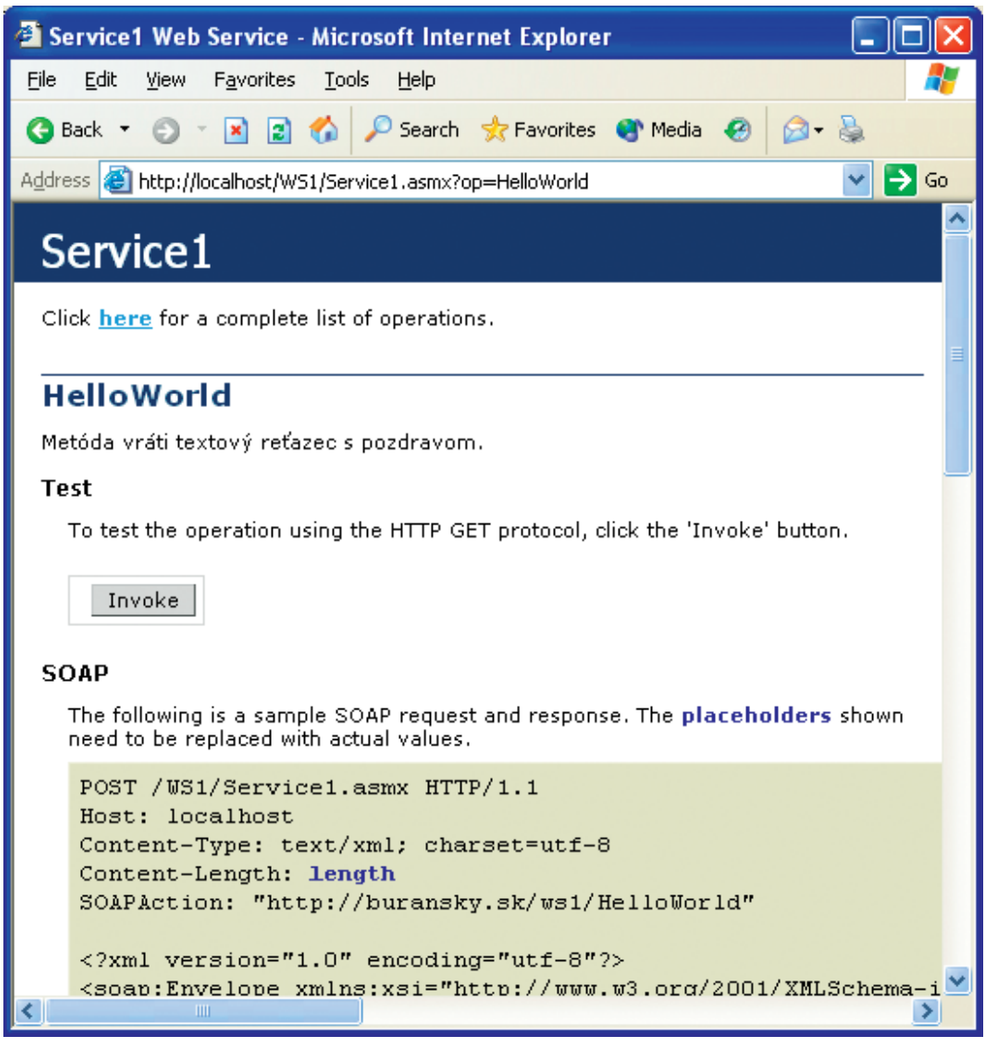
```
[WebMethod(Description="Metóda vráti textový reťazec s pozdravom.")]
public string HelloWorld()
{
  return "Hello World";
}
```

Po vybudovaní získame informácie o webovej službe, ktoré ukazuje obr. 0.2.



Obr. 0.2. Prvá webová služba - po definovaní priestoru mien a opisu služby.

Pokračujme v overovaní a vyvolajme metódu HelloWord. Dostaneme obr. 0.3.

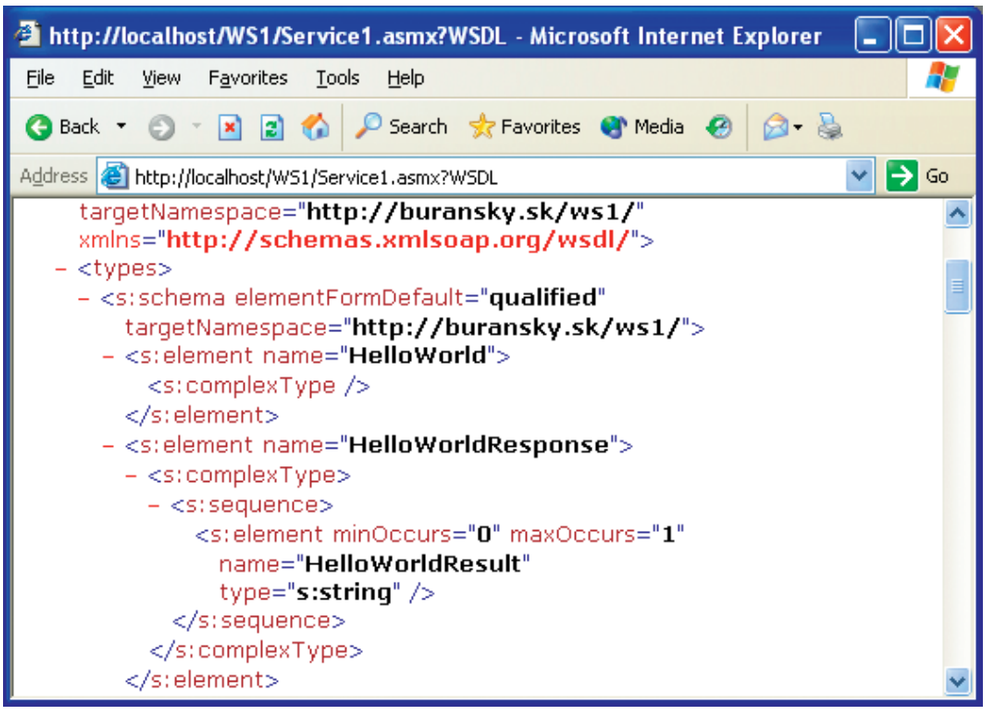


Obr. 0.3. Medzistupeň pri volaní metódy HelloWorld webovej služby.

Po vlastnom vyvolaní (Invoke) služby dostaneme:

```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://buransky.sk/ws1/">Hello World</string>
```

Ak zvolíme Service Description na obr. 0.2, získame opis služby - obr. 0.4.



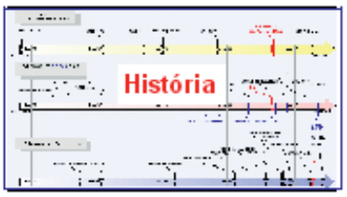
Obr. 0.4. Opis webovej služby.

Z uvedeného opisu tej najjednoduchšej webovej služby, ktorá sa dala vo vývojom prostredí Microsoft VS .NET vytvoriť, vidíme, že máme do činenia so zápismi typu:

```
<Znacka atribut="Hodnota"> obsah </Znacka>
```

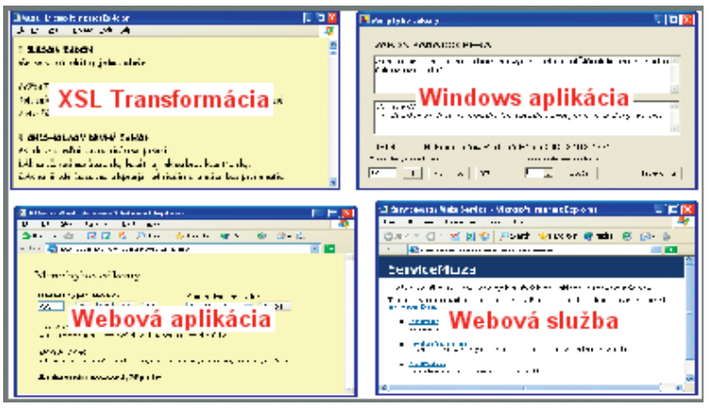
Sú to zápisy v XML. Okrem nich sa stretáme s pojmami WSDL, SOAP, UDDI ...

Cieľom tejto brožúry je ukázať, v čom je podstata XML a súvisiacich technológií, aké miesto majú tieto technológie v tvorbe webových služieb.



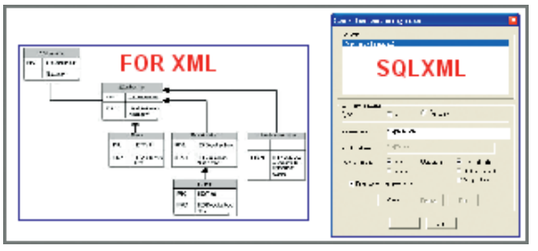
V prvej kapitole je stručný opis histórie značkovacích jazykov. Je ukázaný postup normalizácie aj odraz technológie XML v produktoch firmy Microsoft.

V druhej kapitole je vytvorený XML súbor Murphyho zákonov. Sú objasnené základné pojmy, ako je prvok XML dokumentu, v čom je podstata správného formátovania dokumentu a podstata XML schémy.



V tretej kapitole sú ukážky práce s XML súborom. Je naznačená postata XSL transformácie a použitie XML súboru v aplikácii windows, vo webovej aplikácii aj vo webovej službe.

Vytvorená webová služba je použiá vo webovej aplikácii.



V štvrtej kapitole je opísaná podpora SQL servera pre technológiu XML. Ukázaný je spôsob získavania údajov z relačnej databázy vo formáte XML, použitie šablón a uložených procedúr pre vytvorenie webovej služby s podporou SQLXML 3.0.

V piatej kapitole sú riešené rôzne spôsoby prenosu dát z webovej služby ku klientom. Okrem využitia triedy DataSet je ukázaná aj podpora XML serializácie pre prenos objektov.

V šiestej kapitole sú naznačené možnosti využitia webových služieb v MS Office. Ich využitie umožňuje Microsoft SOAP Toolkit a Office XP Web Services Toolkit.

V siedmej kapitole je načrtnutá budúcnosť webových služieb a charakteristika GXA - Global XML Web Services Architecture.

V dodatku nájdete definície niektorých pojmov, ktoré súvisia s webovými službami.

1. Štipka histórie

Motto: OGDEN-NASHOV ZÁKON
S pokrokom by nič nebolo, len nie a nie ho skončiť.

Plody, ktoré prinášajú značkovacie jazyky, zbiera každý používateľ Internetu. Väčšina z nich si ani neuvedomuje, že úhľadne formátované články s obrázkami a tabuľkami, ktoré vykresľujú internetové prehliadače, sa do ich počítačov dostávajú v podobe textov - HTML dokumentov. Existuje veľa automatizačných prostriedkov, ktoré pomáhajú vytvárať takéto dokumenty. A tak ani autor, ani čitateľ sa o spôsob zápisu dokumentov nemusia starať. HyperText Markup Language je niekde v pozadí. Ak využijeme možnosť, ktorú ponúkajú internetové prehliadače a zobrazíme zdrojový kód, dostaneme sa do sveta značiek. V ostrých zátvorkách nájdeme rôzne údaje, ktoré obklopujú vlastný text. Sú to značky. Vyznačujú začiatok a spravidla aj koniec určitej oblasti dokumentu. Popri vlastných údajoch sú tam aj predpisy na ich zobrazenie (veľkosť a farba textu, odstavce, tabuľky...).

V opísanom scenári bol na oboch stranách človek. Na jednej strane pôvodca dokumentu - autor a na druhej strane čitateľ. Vieme si však predstaviť aj inú situáciu. Zdrojom aj konzumentom údajov môže byť stroj. Človek bude už iba zbierať výsledky, ktoré stroje pripravia. XML má zohrať rozhodujúcu úlohu pri vzájomnej komunikácii počítačov. Existujú vyhlásenia, ktorými je jazyku XML prisudzovaný podobný význam, aký má vynájdenie kníhtlače. V skutočnosti sa však nejedná o žiadnu revolúciu, ale o rozumné zužitkovanie skúseností získaných používaním SGML a HTML.

1.1. SGML

Začiatkom osemdesiatych rokov dvadsiateho storočia IBM, DEC a ďalšie veľké priemyselné firmy prišli k dohode o potrebe štandardu pre výmenu údajov medzi rôznymi počítačmi. Ich iniciatíva viedla k vzniku štandardu ISO (International Organization for Standardization) s číselným označením 8879 z roku 1986. Obsahuje definíciu SGML (Standard Generalized Markup Language). Je to jazyk, ktorého cieľom bolo umožniť zdieľanie informácií medzi podnikmi s rozdielnymi informačnými systémami. Umožňuje oddeliť dáta od ich spracovania. Na základe analýzy štruktúry dát sa vytvára slovník, označovaný ako DTD (Document Type Definition). Tento slovník naznačuje obsah jednotlivých objektov tzv. objektovým modelom s presne definovanou syntaxou. Pretože rôzne množiny údajov môžu mať rôzne dátové objekty, líšia sa aj ich slovníky DTD. Dokument SGML je vytváraný ako textový dokument. Jednotlivé prvky dokumentu sú oddelené značkami, ktoré sú definované v slovníku DTD. Ako text je dokument prenesiteľný na rôzne systémy, ktoré majú implementovaný analyzátor SGML dokumentov. Analyzátor tento dokument načíta, s využitím slovníka DTD a značiek v dokumente je schopný určiť štruktúru dokumentu a spracovať jeho obsah.

Nevýhodou štandardu SGML bolo, že nebol schopný reagovať na požiadavky webu. Vznikol v dobe pomalých a drahých počítačov. Aby bolo možné z týchto systémov získať maximum, štandard SGML bol vybavený minimalizačnými nástrojmi. Tie mali výsledné textové súbory dokumentov stlačiť na čo najmenšiu veľkosť. Dôsledkom boli zložité, drahé a pomalé analyzátory a veľká finančná náročnosť zavádzania SGML do praxe. Skutočnosť, že SGML je štandardom ISO, sa premietla do odmietnutia zmien, ktoré smerovali k jeho zjednodušeniu. Zjednodušenia boli požadované pre možnosť použitia štandardu pri výmenu údajov vo webe a pre možnosť zobrazenia dokumentov prehliadačmi.

1.2. HTML

Namiesto priameho použitia štandardu SGML ako celku veľké rozšírenie získala iba jeho aplikácia HTML - HyperText Markup Language. Pôvodcom jazyka je Tim Berners-Lee. Veľký ohlas mala verzia 2.0, ktorú Berners-Lee a D. Connolly zverejnili v novembri 1995 ako RFC1866. Ďalším medzníkom je január 1997. Bolo zverejnené odporúčanie W3C (World Wide Web Consorcium) HTML 3.2. Ale ani vtedy sa vývoj HTML neskončil. Pokračovalo pridávanie nových značiek. Objavila sa možnosť tvorby skriptov. Doplnené boli kaskádové štýly, formuláre, rámce. Popri statických HTML dokumentoch, čím ďalej tým viac údajov na webových serveroch sa ukladá v databázach. Čím ďalej tým viac dokumentov sa vytvára dynamicky programami alebo skriptami s využitím údajov v databázach. Rozširuje sa eBusines, eLearning, eBanking. Narastajú požiadavky na výmenu údajov medzi strojmi...

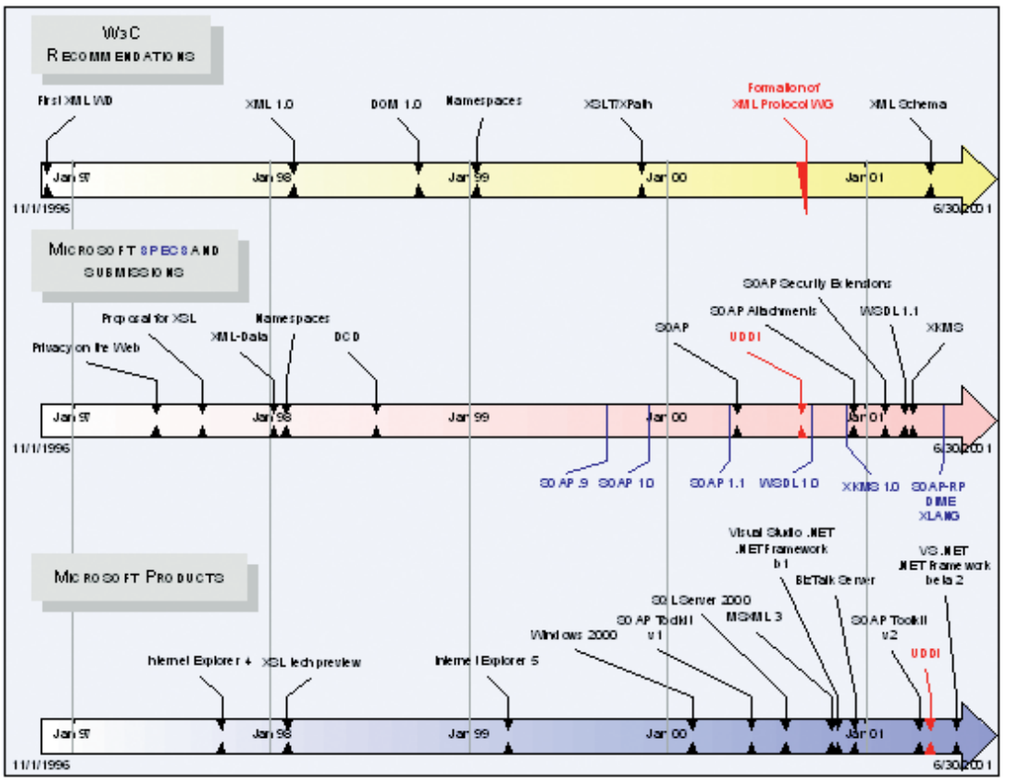
HTML bolo pôvodne určené pre prezentáciu textov. Veď je to vyjadrené aj v názve - hypertext. V textoch sa dá vyjadriť všetko. Tak sú písané aj knihy. Človek, čitateľ si z kníh aj zobrazených webových stránok potrebnú informáciu vyberie. Aby to však mohli robiť aj stroje, je potrebné dôslednejšie formátovať údaje a treba mať jasné pravidlá, podľa ktorých je možné zistiť, či sú údaje úplné, aby sa „nepoplietli slivky s hruškami“. HTML k tomu nestačí, SGML je zložité, ISO neústupčivé, a tak prichádza na scénu XML.

1.3. Vznik XML

V roku 1996 bola pod záštitou W3C vytvorená pracovná skupina, ktorej cieľom malo byť zjednodušenie SGML pre potreby webu. Okrem širokej akceptovateľnosti výsledného štandardu pre potreby webu sa mala dosiahnuť aj možnosť jednoduchej tvorby programov pre spracovanie XML dokumentov. Prvá verzia odporúčania XML bola zverejnená vo februári 1998. V októbri roku 2000 bola zverejnená revízia tohto odporúčania pod názvom **Extensible Markup Language (XML) 1.0 (Second Edition)**.

Odporúčanie definuje, čo je to XML dokument, čo je prvok (element), jeho počiatočné a koncové ohraničenie, značka, atribúty aj obsah prvku. Určuje pravidlá pre voľbu názvov prvkov - značiek, atribútov. Stanovuje tiež, kedy je dokument dobre sformovaný (well-formed - niekedy prekladané tiež správne sformátovaný) a tiež kedy je dokument platný (valid).

Prijatie odporúčania W3C o XML spustilo lavínu ďalších aktivít. Firma Microsoft stála pri zrode XML a vynaložila veľkú úsilie pre rozšírenie technológie XML tak, aby bola využiteľná pri tvorbe webových služieb. Ukazuje to aj obr. 1.1. Je na ňom zachytená časová postupnosť prijatia rozhodujúcich odporúčaní W3C, návrhov zo strany firmy Microsoft, ako aj vypustenia produktov firmy Microsoft, ktoré využívajú a podporujú XML a súvisiacich technológií.



Obr. 1.1. Aktivity firmy Microsoft pri presadzovaní technológie XML.

2. Vytvorme XML dokument

Motto: ILESOV ZÁKON
Všetko sa dá robiť aj jednoduchšie.

Najrýchlejšia cesta k pochopeniu podstaty XML je preskúmať zopár takých dokumentov. Stačí pohľadať súbory s príponou xml (ozaj, koľko ich máte na svojom počítači?) a niektoré z nich zobraziť napr. v Notepade alebo v Internet Exploreri. Uvidíme ostré zátvorky, v nich nejaké názvy, atribúty, texty... Po prezretí takých dokumentov možno dostanete nápad - vytvoriť si svoj vlastný XML dokument. Nie vždy sa totiž dá ľahko zistiť, čo je zmyslom jednotlivých XML dokumentov, ktoré nájdeme vo svojom počítači. Poďme si teda nejaký vytvoriť.

2.1. Hľadanie námetu

Pri hľadaní námetu pre príklady sme siahli po Murphyho zákonoch. Je to dané tým, že tieto zákony sú dostatočne známe. Občas sa oplatí zopakovať si ich, aby sme pochopili, prečo sa veci kazia. Treba len dúfať, že práca s nimi vo forme XML nám nepokazí náladu.
Vytvorme XML dokument, ktorý bude obsahovať Murphyho zákony. Ukážme si konkrétny príklad zápisu takých zákonov:

Murphyho zákony

ROBERTOVA AXIÓMA
Existujú iba omyly.

ILESOV ZÁKON
Všetko sa dá robiť aj jednoduchšie.

MALEKOV ZÁKON
Najjednoduchšie myšlienky sa najzložitejšie vyjadrujú.

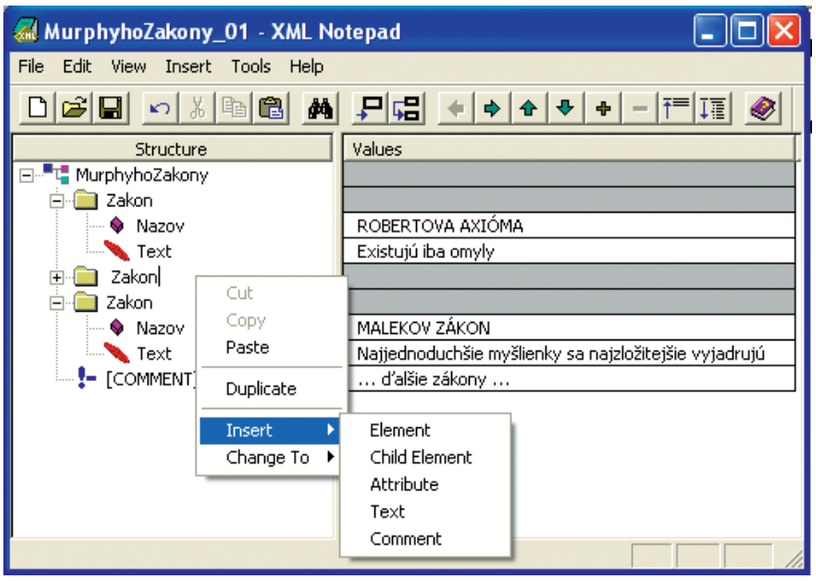
atď. ďalšie zákony.

Čo vieme povedať o uvedenom zápise? Je tam nadpis, ktorý nás informuje o tom, že nasledujú Murphyho zákony. Ak vidíme text ROBERTOVA AXIÓMA resp. ILESOV ZÁKON, „domyslíme“ si, že je to názov zákona. Za ním nasleduje vlastný text zákona. Odstavce oddeľujú zákony. O zákone sa teda dá hovoriť vtedy, ak poznáme jeho názov a vlastný text zákona.

2.2. Použitie XML Notepadu

Pre zápis zákonov do XML dokumentu využijeme XML Notepad. V júli 2002 bolo možné získať XML Notepad Beta 1.5 Installer (xpsetup.exe; 335K) na adrese:
http://msdn.microsoft.com/library/en-us/dnxml/html/xpsetup.exe

Použitie XML Notepad ukazuje obr. 2.1.

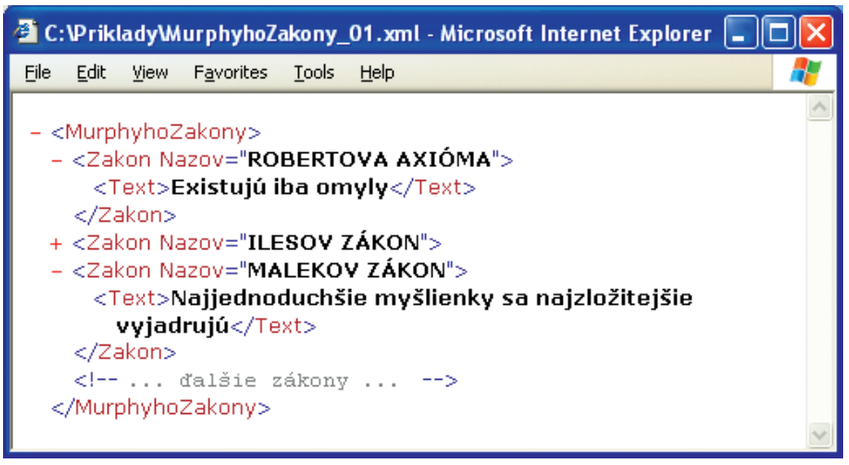


Obr. 2.1. Použitie XML Notepadu pre vytvorenie dokumentu.

Je to veľmi jednoducho ovládateľný nástroj. Jeho pracovná plocha je rozdelená na dve časti. V ľavej sa zobrazuje štruktúra tvoreného dokumentu a v pravej hodnoty. V zobrazenej štruktúre dokumentu Murphyho zákonov vidieť, že pri niektorých vrcholoch je znamienko mínus a pri niektorých znamienko plus. V druhom prípade sa jedná o vrcholy, ktoré je možné rozvinúť (kliknutím na +) a vnárať sa tak do hĺbky k ďalším podrobnostiam. Časť rozvinutého dokumentu, ktorý nepotrebujeme vidieť, možno skryť (kliknutím na -). Kontextové menu umožňuje vložiť nový prvok (element), atribút, text resp. komentár. Jednoducho môžeme dokument rozširovať o nové prvky a zadávať ich hodnoty. Výsledok sa dá uložiť do textového súboru (spravidla s rozšírením - extension xml). Získame tak XML dokument.
V našom prípade to bude dokument s názvom MurphyhoZakony_01.xml.
Treba poznamenať, že XML Notepad nie je jediným prostriedkom pre tvorbu XML dokumentov. Veľké možnosti pre prácu s XML dokumentami ponúka Microsoft Visual Studio .NET. K ich využitiu sa ešte dostaneme.

2.3. Zobrazenie Internet Explorerom a základné pojmy

Internet Explorer zobrazí vytvorený súbor tak, ako ukazuje obr. 2.2. Aj tam vidieť na začiatkoch riadkov znaky mínus a plus. Podobne ako XML Notepad, aj Internet Explorer umožňuje rozvinúť resp. skryť časti XML dokumentu. Štruktúra dokumentu je tu vyjadrená červenou farbou. Zvýrazneným čiernym písmom sú uvedené hodnoty tvoriace informačný obsah dokumentu. Modrá farba bola prisúdená riadiacim znakom. Texty komentárov sú šedej farby.



Obr. 2.2. Zobrazenie XML dokumentu Internet Explorerom.

Už letmým pohľadom na obr. 2.2. zistíme výskyt takýchto prvkov (elementov):

```
<Značka atribút1="hodnota1" atribút2="hodnota2">obsah  prvku </Značka>
```

Medzi počiatočným a koncovým ohraničením prvku je uvedený jeho obsah. Počiatočné aj koncové ohraničenie je v ostrých zátvorkách.

Označenie nahradzujúce slovo Značka je názov prvku. V našom XML dokumente sú za názvy prvkov zvolené texty MurphyhoZakony, Zakon a Text. Sú určité obmedzenia na voľbu názvov prvkov. Tak napríklad je stanovené, že názvy prvkov sa musia začínať písmenom, nie sú v nich povolené medzery a niektoré špeciálne znaky (urobíme dobre ak sa vyhneme znakom s diakritikou). V názvoch sa rozlišujú veľké a malé písmená. Tak napr. Zakon a ZAKON sú považované za rôzne názvy.

V počiatočnej značke môžu byť uvedené atribúty prvku s priradenou hodnotou v úvodzovkách. Označenie nahradzujúce slovo atribút1, atribút2 (vo všeobecnosti atributi) je názov atribútu. Na voľbu názvov atribútov sa vzťahujú tie isté obmedzenia ako na voľbu názvov prvkov. V našom XML dokumente prvok Zakon má jeden atribút - Nazov. Za znakom priradenia '=' je v úvodzovkách uvedená hodnota daného atribútu (namiesto úvodzoviek je možné použiť aj apostrofy).

Obsahom prvku môže byť hodnota, ale aj jeden i viac prvkov. Prvky Text v našom dokumente obsahujú „čisté“ hodnoty - nie sú v nich „vnorené“ ďalšie prvky. Obsahom ostatných prvkov sú prvky. Takýmto spôsobom sa vytvára stromová štruktúra. V jednom koreňovom prvku (root) sú v norené ostatné prvky. Koreňovým prvkom nášho dokumentu je prvok MurphyhoZakony. Ten obsahuje jeden alebo viac prvkov Zakon. Zakon obsahuje prvok Text.

V XML dokumente môžu byť poznámky. V našom dokumente je poznámka využitá na označenie miesta, kde treba vložiť ďalšie zákony:

```
<!-- ... ďalšie zákony ...-->
```

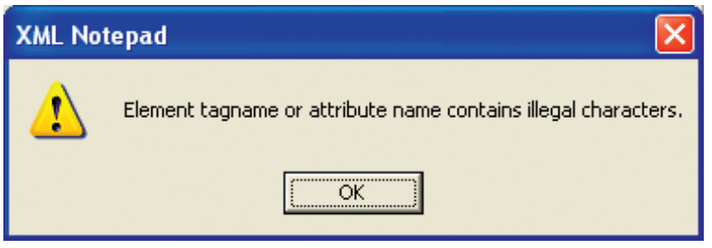
Znaky `<!--` označujú začiatok a znaky `-->` koniec komentára.

Sú stanovené pravidlá, pomocou ktorých sa dá vyhodnotiť, či je dokument dobre sformátovaný (well-formed). Patria k nim už spomenuté pravidlá pre voľbu názvov prvkov a atribútov, či spôsob zadávania hodnôt atribútov (úvodzovky, apostrofy). Okrem toho je stanovené, že XML dokument musí mať jediný koreňový prvok. Každej počiatočnej značke musí zodpovedať koncová značka. Nie je prípustné „kríženie“ značiek, napr.:

```
<A> obsah A  <B> obsah B  </A>  pokračovanie obsahu B  </B>
```

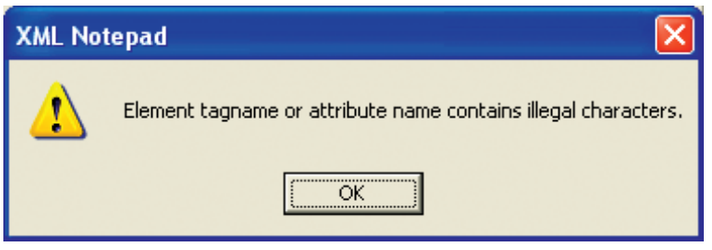
V uvedenom príklade by prvok B čiastočne patril do obsahu prvku A, ale jeho pokračovanie by už bolo mimo prvku A. To prípustné nie je.

V prípade, že XML dokument tvoríme pomocou nástroja podobnému XML Notepad, máme uľahčenú úlohu. Mnohé zápisy, ktoré by viedli k nesprávnemu sformátovaniu dokumentu, nie sú umožnené. XML Notepad sa stará o uzatvorenie prvku koncovou značkou. Upozorní tiež na chyby pri pomenovaní prvkov a atribútov. Tak napríklad, ak by sme pomenovali koreňový prvok nášho dokumentu Murphyho Zakony (medzi slovami je medzera), dostali by sme upozornenie, ktoré je na obr. 2.3.



Obr. 2.3.: Oznam XML Notepadu o neprípustnom znaku v názve.

XML dokument je textový súbor. Ako taký ho možno napísať v textovom editore. Strácame však „strážcu“, ktorý dozerá na správnosť sformátovania dokumentu. A tak sa môže stať, že „vyrobíme“ nejakú chybu. Zobrazením dokumentu v internetovom prehliadači, ktorý má zabudovaný analyzátor XML dokumentov (Internet Explorer ho má od verzie 4.0), získame informáciu o tom, či je dokument dobre sformátovaný. Ak ho prehliadač zobrazí v podobe, akú ukazuje obr. 2.2., dokument je dobre sformátovaný. V prípade chyby môžeme získať výpis, ktorý ukazuje obr. 2.4.



Obr. 2.4. Oznam Internet Explorera o nesprávnom sformovaní dokumentu.

Chybové hlásenie na obr. 2.4. vzniklo po tom, keď koncová značka prvku Zakon bola napísaná veľkými písmenami. Takú chybu XML Notepad nedovolí urobiť, lebo robí automatický zápis koncovej značky. Neumožní ani „prekríženie“ prvkov dokumentu. Také chyby však môžeme urobiť, ak editujeme XML dokument v obyčajnom textovom editore (kto by to však robil).

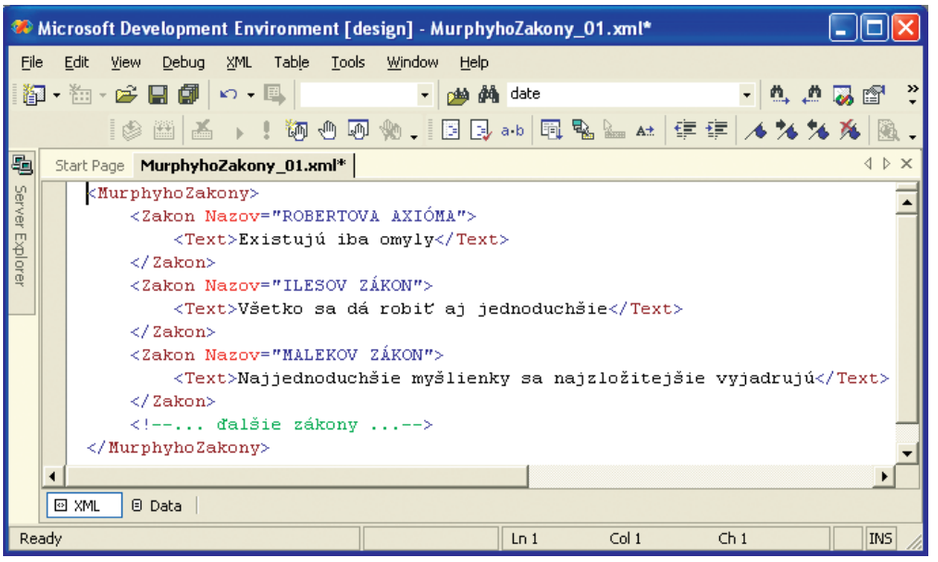
Problematike XML je v poslednej dobe venovaných veľa publikácií. Ten, kto sa nechce spoľahnúť na sprostredkované informácie, siahne po originálnom znení odporúčania W3C. V ňom sa uvádza aj to, že súčasťou každého XML dokumentu by mala byť aj informácia o verzii XML špecifikácie aj informácia o použitom kódovaní. Prvým riadkom dokumentu by mal byť tzv. prológ, napr.:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

XML Notepad je užitočný pomocník, ktorý uľahčuje tvorbu XML dokumentov. Väčšie možnosti pre prácu s XML však ponúka Microsoft Visual Studio .Net. Medzi nástrojmi, ktoré toto prostredie obsahuje, je aj XML Designer.

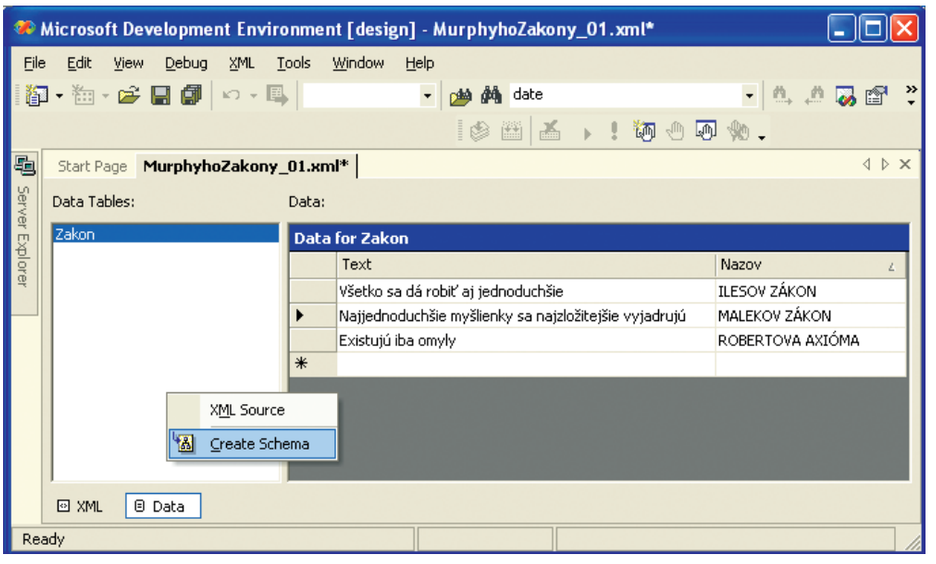
2.4. XML Designer v MS Visual Studio .Net

Pohľad na náš dokument, ktorý ponúka XML Designer vo vývojovom prostredí MS Visual Studio .Net, ukazuje obr. 2.5. a 2.6.



Obr. 2.5. Pohľad na XML súbor prostredníctvom MS Visual Studio .NET - pohľad XML.

Prepínanie medzi dvoma spôsobmi zobrazenia XML dokumentu sa dosiahne pomocou záložiek XML resp. Data v spodnej časti okna.



Obr. 2.6. Pohľad na XML súbor prostredníctvom MS Visual Studio .NET - pohľad Data.

Ak porovnáme obr. 2.5 a obr. 2.6, zistíme rozdielne poradie zákonov. Jednoduchým kliknutím na hlavičku v tabuľke zobrazených dát (v pohľade Data - obr. 2.6) získame usporiadanie podľa príslušného stĺpca. Obr. 2.6 zachytáva usporiadanie podľa stĺpca Nazov.

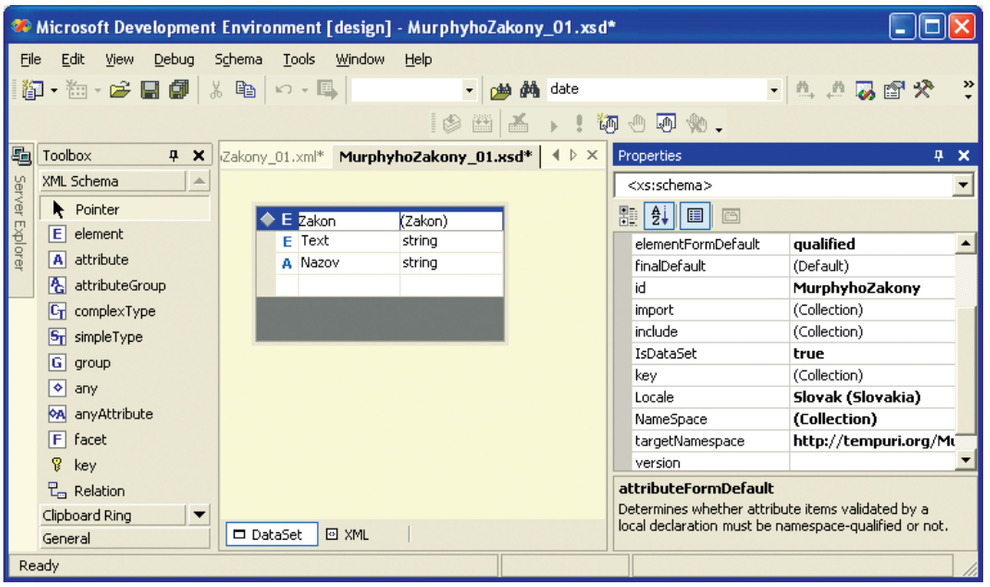
Na obr. 2.6. si treba všimnúť kontextové menu, ktoré ponúka možnosť vytvorenia schémy pre náš XML dokument. Ak túto možnosť využijeme, XML Designer upraví počiatočné ohraničenie koreňového prvku nášho dokumentu takto:

```
<MurphyhoZakony xmlns="http://tempuri.org/MurphyhoZakony_01.xsd">
```

Doplní atribút xmlns a jeho hodnotu. Ako uvidíme neskôr, ponúknutú hodnotu identifikujúcu priestor mien nášho dokumentu (XML Namespace) bude možné zmeniť. Najskôr sa však pozrime na to, čo za schému sme to získali.

2.5. XML schéma

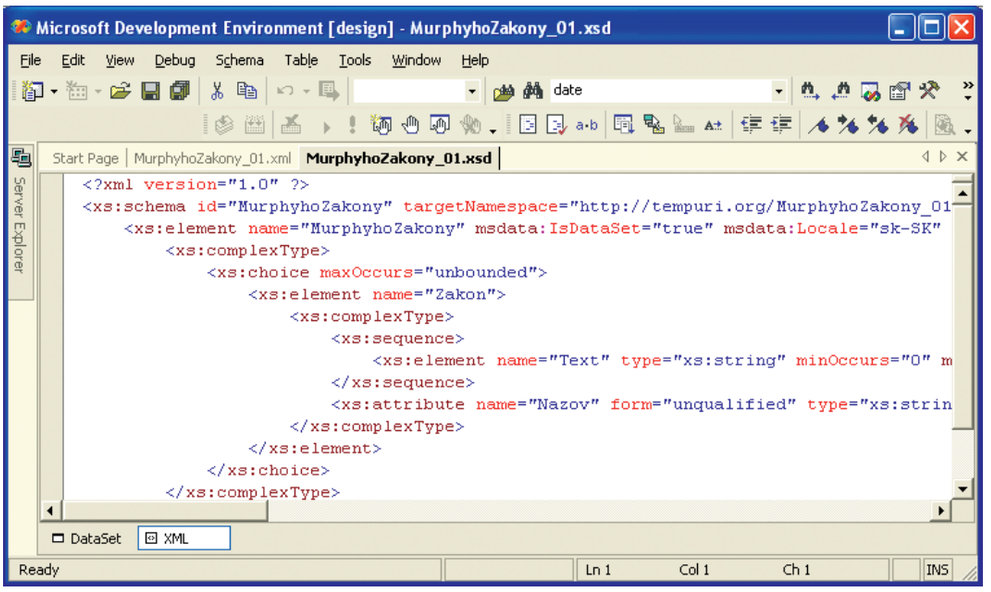
Použitím príkazu Create Schema XML Designer pre náš dokument vytvoril XML schému. Ukazuje ju obr. 2.7.



Obr. 2.7. XML schéma pre Murphyho zákony - pohľad DataSet.

V pracovnom priestore sú tri okná - paleta nástrojov (Toolbox), okno pohľadu na XML schému a okno vlastností (Properties). V okne pohľadu vidíme, že v našom dokumente je prvok (element) Zakon (E v hlavičke tabuľky pred slovom Zakon). V tabuľke je uvedené, čo môže byť obsahom prvku. Je tam atribút Nazov a prvok Text (A resp. E v prvom stĺpci tabuľky). Text aj názov je typu string (posledný stĺpec tabuľky). Z uvedeného vidieť, že XML schéma opisuje usporiadanie súčastí XML dokumentu a definuje ich typy.

Z palety nástrojov sa dajú ponúknuté komponenty umiestniť v okne pohľadu a tak schému rozširovať. V okne vlastností je možné definovať vlastnosti prvku, ktorý je označený v okne pohľadu na XML schému. V spodnej časti vidieť, že sa dá zvoliť aj pohľad XML. Zachytáva to obr. 2.8. (paleta nástrojov a okno vlastností sú skryté).



Obr. 2.8. XML schéma pre Murphyho zákony - pohľad XML.

Vidieť, že vlastná schéma XML dokumentu, ktorá definuje jeho typy, je XML dokumentom. Treba poznamenať, že v odporúčaní W3C pre XML sú špecifikované pravidlá definovania typov dokumentu (DTD - Document Type Definition). Tie vychádzajú z SGML a stali sa terčom kritiky, lebo syntax DTD nezodpovedá syntaxi XML. Zástupcovia firmy Microsoft predložili združeniu W3C návrh XML Data, ktorý bol zverejnený už v januári 1998. Jeho redukovaná verzia XDR (XML Data Reduced) sa stala súčasťou Internet Explorera 5. Združenie W3C reagovalo na kritiku zahájením prác na alternatívnom spôsobe definovania typu dokumentu. Vyústili do odporúčania **XML Schema**, ktoré bolo zverejnené v máji 2001. Vývojové prostredie MS Visual Studio .NET podporuje už iba XML Schema. Je možné očakávať, že DTD a XDR sa postupne prestanú používať.

Pridružením deklarácie typov k XML dokumentu sa dosiahne možnosť uskutočniť kontrolu dokumentu. Dobre sformátovaný XML dokument, ktorý má pridruženú deklaráciu typu dokumentu a vyhovuje všetkým v nej definovaným obmedzeniam, je platný (valid). Deklarácia typu dokumentu teda oznamuje analyzátoru (napríklad Internet Exploreru), kde nájde pravidlá, voči ktorým musí celý dokument preveriť. Poznamenajme, že pod pojmom analyzátor tu rozumieme to, čo sa zvykne označovať aj slovom parser.

Pri pohľade na XML schému (obr. 2.8) si všimnite, že názvy prvkov začínajú trojicou znakov „**xs:**“. Vyjadruje sa tým, že sa používa priestor mien (Namespace) xs. Treba poznamenať, že W3C prijalo odporúčanie **Namespaces in XML** už v januári 1999. Jeho cieľom je riešiť možné konflikty pri voľbe názvov prvkov a atribútov.

2.6. Tvorba XML schémy

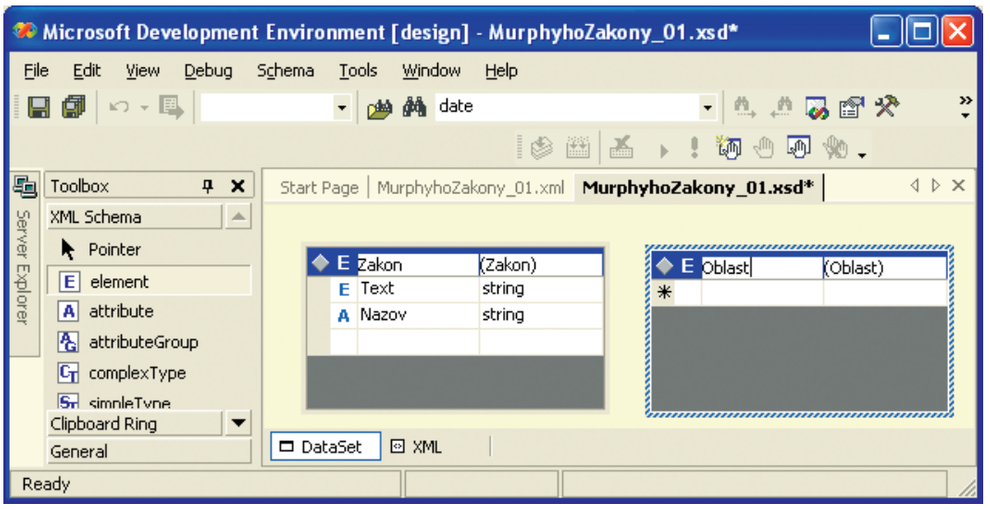
Schému, na ktorú sme sa doteraz pozerali, sme získali až po vytvorení XML súboru. Treba poznamenať, že tento postup bol zvolený preto, aby sme predstavili základné pojmy, ktoré súvisia s technológiou XML. Preto sme siahli po jednoduchom nástroji, akým je XML Notepad. Ním sa dá dosiahnuť správné sformátovanie XML dokumentu. Oveľa častejšie, než k tvorbe XML dokumentu sa programátor dostáva k riešeniu využitia XML dokumentov. Ako uvidíme neskôr, XML dokumenty sa veľmi jednoducho dajú získať napr. zo záznamov v databáze. Nehľadiac na to, zostaneme ešte chvíľu v pozícii, že „chceme“ vytvoriť svoj vlastný XML dokument.

Doteraz sme uvažovali o veľmi jednoduchej štruktúre XML dokumentu pre záznam Murphyho zákonov. Trochu ju skomplikujeme, aby sme mohli ukázať upresňovanie XML schémy. Vytvoríme si tak aj predpoklady pre ďalšie ukážky práce s XML súbormi.

Doplňujúce požiadavky na XML dokument pre záznam Murphyho zákonov budú tieto:

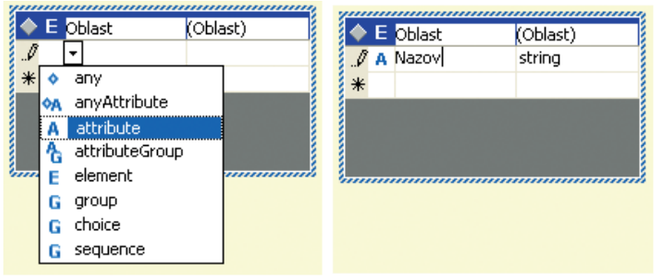
1. Zaraďovať zákony do oblastí. Oblasť je určená svojím názvom.
2. Zákon treba vedieť jednoznačne identifikovať - zaviesť k tomu identifikátor.
3. Zákon môže obsahovať viac odstavcov - jednotiek textu.
4. Zákon môže obsahovať jeden alebo viac dodatkov.
5. Zákon má priradené hodnotenie. Čitateľ má možnosť priradiť svoje subjektívne hodnotenie zákona známkou v stupnici 1 až 5, pričom 1 je najlepšie a 5 najhoršie hodnotenie (podobne ako v škole). Zaznamenáva sa priemerné hodnotenie, počet hodnotení, čas posledného hodnotenia.

Pri úprave XML schémy budeme vychádzať zo situácie, ktorú zachytáva obr. 2.7. V okne Toolbox označíme element a pri zatlačenom pravom tlačidle myši umiestnime nový prvok v okne DataSet. Ponúknutý názov element1 zmeníme na Oblast. Výsledok ukazuje obr. 2.9.



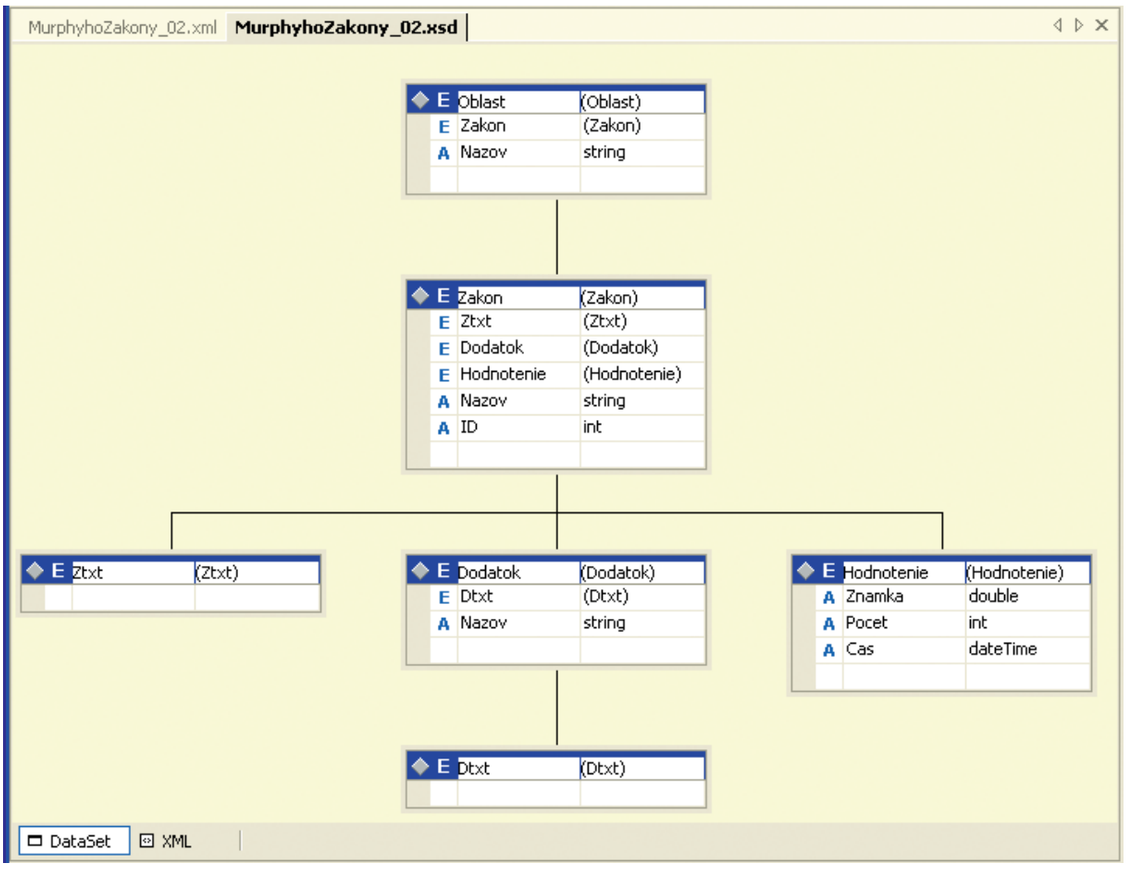
Obr. 2.9. Tvorba XML schémy - doplnenie prvku Oblast.

V prvom stĺpci v tabuľke prvku Oblast v riadku označenom hviezdičkou klikneme ľavým tlačidlom myši. Kliknutím na malé tlačidlo so šípkou, ktoré sa objaví, vyvoláme kontextové menu. Situáciu zachytáva obr. 2.10 vo svojej ľavej časti. Zvolíme attribute. Ponúknutý názov attribute1 prepíšeme na Nazov. Ponecháme ponúknutý typ atribútu - string. Situáciu zachytáva obr. 2.10 vo svojej pravej časti.



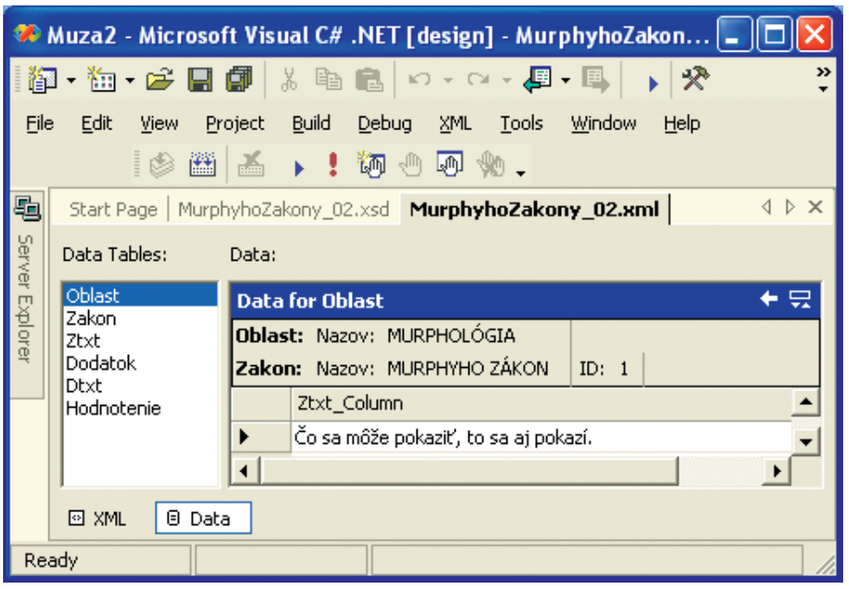
Obr. 2.10. Tvorba XML schémy - doplnenie atribútu Nazov prvku Oblast.

Načrtnutým spôsobom môžeme rozširovať XML schému tak, aby sme dosiahli štruktúru XML dokumentu, ktorá vyhovuje vyššie definovaným požiadavkám. Vytvoríme tak ešte elementy Dodatok, Hodnotenie, Ztxt, Dtxt a v nich potrebné atribúty. Metódou „drag and drop“ vytvorené elementy zoskupíme tak, že dostaneme výslednú schému, ktorú zachytáva obr. 2.11.



Obr. 2.11. Rozšírená XML schéma pre Murphyho zákony.

XML Designer vývojového prostredia VS .NET umožňuje pracovať s XML súborom, ktorý zodpovedá takto vytvorenej schéme. Ukazuje to obr. 2.12.



Obr. 2.12. Práca s XML dokumentom Murphyho zákonov po rozšírení XML schémy.

Ak by sme mali dostatok trpezlivosti, mohli by sme „ručne“ zapísať do takto pripraveného XML dokumentu Murphyho zákony. Je to však veľmi nevďačná úloha. Namiesto toho v nasledujúcej kapitole ukážeme, ako tento proces zautomatizovať.

2.7. Prepis textu do XML súboru

Majme Murphyho zákony v textovom súbore v tvare, ktorý ukazuje tab. 2.1.

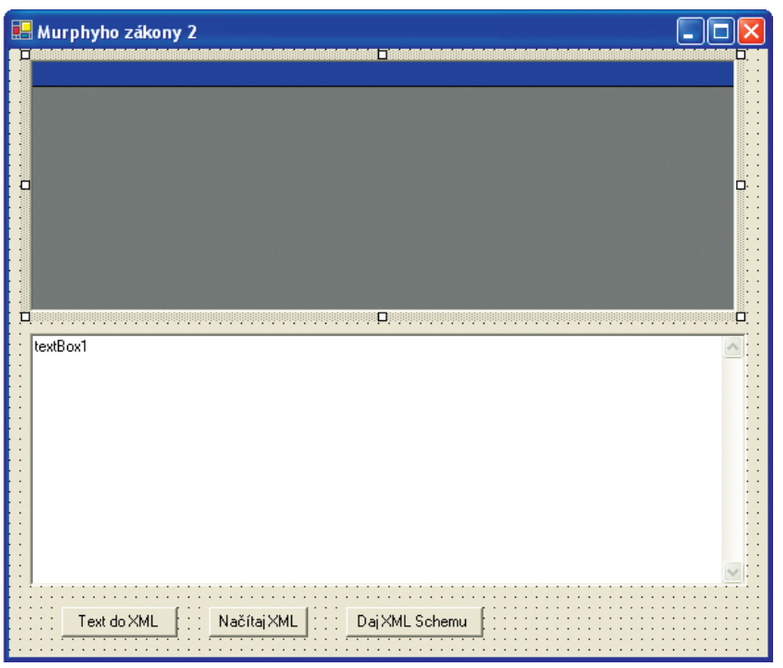
Tab. 2.1. Tvar zápisu Murphyho zákonov v textovom súbore, dr - druh riadku.

dr	Obsah textového súboru
1	//
2	Názov oblasti
3	\
4	Názov zákona
4	Text zákona
4	Ďalší text zákona
..	....
5	+
6	Názov dodatku
7	Text dodatku
7	Ďalší text dodatku
3	\
..	....

Obsah textového súboru je v druhom stĺpci tabuľky. Prvý stĺpec je použitý iba pre označenie druhu riadku v textovom súbore. Vidieť, že obsah textového súboru je zostavený podľa určitých pravidiel. Sú v ňom riadky s riadiacimi znakmi, ktoré označujú začiatok oblasti (dr=1), zákona (dr=3), dodatku (dr = 5). Potom nasleduje riadok s názvom oblasti, zákona resp. dodatku. Za názvom zákona nasleduje jeden alebo aj viac riadkov s textom zákona (dr=4). Zákon môže (ale nemusí) mať jeden ale aj viac dodatkov. Za názvom dodatku nasleduje jeden alebo aj viac riadkov s textom dodatku (dr=7).

Celkom rozumne zostavený textový súbor so zaujímavým obsahom. Vytvorme z neho XML dokument.

V Microsoft Visual Studio .Net vytvorme nový projekt typu C# so šablónou Windows Application. Nazvime ho napr. Muza2. V návrhovom zobrazení začleňme do formulára aplikácie tri tlačidlá (button), data grid, a text box, ako ukazuje obr. 2.13.



Obr. 2.13. Formulár aplikácie Muza2.

V obsluhe stlačenia tlačidla **Text do XML** vytvoríme transformáciu textového súboru do XML súboru. Vytvorený dokument načítame do data gridu pri stlačení tlačidla **Načítaj XML**. Do textového poľa vypíšeme XML schému dokumentu v obsluhe stlačenia tlačidla **Daj XML Schému.**

Pre vyriešenie transformácie textového súboru Murphyho zákonov do XML súboru je rozumné využiť objekt triedy **XmlTextWriter** z priestoru mien System.XML. Táto trieda ponúka viaceré užitočné metódy pre zápis do súboru. Pre riešenie našej úlohy využijeme metódy:

- **WriteStartDocument** .. zápis XML definície do súboru s verziou „1.0“,
- **WriteStartElement** .. zápis počiatočnej značky prvku,
- **WriteAttributeString** .. zápis atribútu - názvu aj hodnoty,
- **WriteString** .. zápis textu - obsahu prvku,
- **WriteEndElement** .. zápis koncového ohraničenia prvku,
- **WriteEndDocument** .. zápis koncového ohraničenia pre všetky otvorené prvky.

Poznanie možností triedy XmlTextWriter, najmä metód, ktoré sú tu uvedené, vytvára predpoklady, že vyriešenie našej úlohy je „rutinnou“ záležitosťou. Pre zaujímavosť opíšeme jedno z možných riešení - kód v zdrojovom súbore Form1.cs našej aplikácie.

Doplňme použitie týchto priestorov mien:

```
using System.IO;
using System.Xml;
using System.Text;
using System.Globalization;
```

Definujme vymenúvací typ a deklarujme členské premenné triedy Form1:

```
// Vymenúvací typ EStav bude slúžiť pre zaznamenanie stavu
// spracovania textového súboru
enum EStav {Zaciatok, Oblast, Zakon, Dodatok };
private EStav Stav=EStav.Zaciatok; // pre zaznamenanie stavu spracovania
private EStav NovyStav;            // pre poznačenie nového stavu
private String element="";         // element, ktorý sa má začať písať
private String teraz="";           // časový údaj v textovej podobe
private int IDZakon=0;             // identifikátor zákona
```

V konštruktore triedy Form1 nastavme hodnotu premennej teraz takto:

```
teraz = DateTime.Now.ToString("s", DateTimeFormatInfo.InvariantInfo);
```

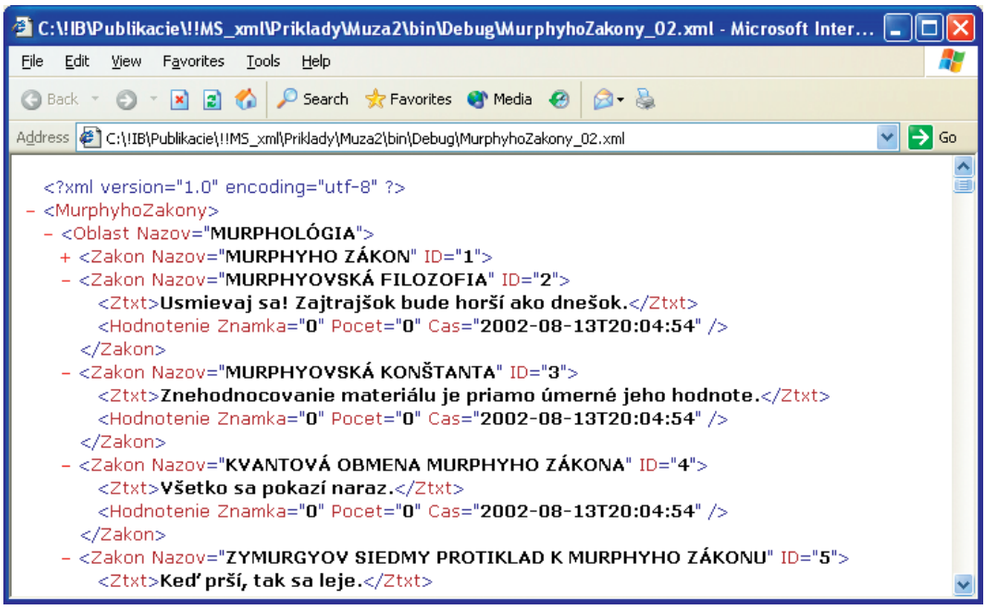
Definujme obsluhu tlačidla Text do XML - **ButTextDoXML_Click** a pomocné členské metódy triedy Form1 **ZapisKoncoveOhranicenia** a **ZapisHodnotenie**:

```
private void ButTextDoXML_Click(object sender, System.EventArgs e)
{ // IB **********************
  // Vytvorenie XML súbor z textového
  String text="";      // riadok textu načítaný zo súboru
  StreamReader stream = new   StreamReader("MurphyhoZakonyU.txt",System.Text.Encoding.UTF8,
false);
  XmlTextWriter xwriter = new XmlTextWriter("MurphyhoZakony_02.xml",System.Text.Encoding.UTF8);

 // Nastavím formátovanie pre čitateľnosť výsledného súboru,
  // budú zapisované konce riadkov a odsadzovanie textu.
  xwriter.Formatting = Formatting.Indented;
  // Zápis začiatku dokumentu
  xwriter.WriteStartDocument();
  // Zápis počiatočného ohraničenia prvku "MurphyhoZakony".
  xwriter.WriteStartElement("MurphyhoZakony");
```

```
  // Cyklus čítania z textového súboru a zápis do XML súboru
  while (stream.Peek() > -1)
  { // Načítanie ďalšieho riadku textu a jeho vyhodnotenie
    text= stream.ReadLine();
    ZapisKoncoveOhranicenia(text, xwriter);
    if (NovyStav!=EStav.Zaciatok) {
      // bol načítaný riadiaci riadok - načítam ďalší .. Názov
      text= stream.ReadLine();
      if ( element.Length>0 && text.Length>0) {
        xwriter.WriteStartElement(element);
        // Zápis atribútu Názov
        xwriter.WriteAttributeString("Nazov", text);
        // Pre Zákon zapíšem aj identifikátor
        if (NovyStav==EStav.Zakon) {
          IDZakon++; // najskôr ho inkrementujem
          xwriter.WriteAttributeString("ID", IDZakon.ToString());
        }
        // akceptujem nový stav
        Stav=NovyStav;
      }
    } else {
      // Zápis načítaného riadku - ak sa zapisuje Zákon alebo jeho Dodatok
      switch (Stav)  {
        case EStav.Dodatok:
          xwriter.WriteStartElement("Dtxt");
          xwriter.WriteString(text);
          xwriter.WriteEndElement();
          break;
        case EStav.Zakon:
          xwriter.WriteStartElement("Ztxt");
          xwriter.WriteString(text);
          xwriter.WriteEndElement();
        break;
      }
    }
  }
  ZapisKoncoveOhranicenia("///", xwriter);

  // Koncové ohraničenie prvku "MurphyhoZakony".
  xwriter.WriteEndElement();

  // Koniec dokumentu .. zatvorí všetky otvorené elementy
  xwriter.WriteEndDocument();

  // Zatvorím XML dokument
  xwriter.Close();

  // Oznámim počet zapísaných zákonov
  text="Pocet zapísaných zákonov: "+IDZakon.ToString();
  MessageBox.Show(text);
}

private void ZapisKoncoveOhranicenia(String text, XmlTextWriter xwriter)
{ // IB ***************************
  // V závislosti od načítaného riadku textu zapíše do XML
  // súboru koncové ohraničenia. Tie závisia tiež od stavu
  // transformácie, t.j. čo sa práve zapisovalo.
  // Okrem zápisu koncových ohraničení nastaví členské premenné:
  // NovyStav .. aký je novy stav - ak text obsahuje riadiaci znak,
  // element  .. názov prvku, ktorý bude zapisovaný
```

```
  NovyStav=EStav.Zaciatok;
  if (text.Length>0)
  {
    switch (text[0])
    { case '/':   // Začína sa nová oblasť
        NovyStav=EStav.Oblast;
        element="Oblast";
        switch (Stav)
        { // Uzatvorím doteraz rozpísané elementy
          case EStav.Dodatok:
            xwriter.WriteEndElement();
            goto case EStav.Zakon;
          case EStav.Zakon:
            ZapisHodnotenie (xwriter);
            goto case EStav.Oblast;
          case EStav.Oblast:
            xwriter.WriteEndElement();
            break;
         }
         break;
      case '\\':  // Začína sa nový zákon
        NovyStav=EStav.Zakon;
        element="Zakon";
        switch (Stav)
        { // Uzatvorím doteraz rozpísané elementy
          case EStav.Dodatok:
            xwriter.WriteEndElement();
            goto case EStav.Zakon;
          case EStav.Zakon:
            ZapisHodnotenie (xwriter);
            break;
          case EStav.Oblast: break;
          default : element=""; break;
        }
        break;
      case '+':   // Začína sa nový Dodatok
        NovyStav=EStav.Dodatok;
        element="Dodatok";
        switch (Stav)
        { // Uzatvorím doteraz rozpísane elementy
          case EStav.Dodatok: xwriter.WriteEndElement();break;
          case EStav.Zakon: break;
          case EStav.Oblast: break;
          default : element=""; break;
        }
        break;
    }
  }
}

private void ZapisHodnotenie (XmlTextWriter xwriter)
{ // IB ********************
  // Zapíše hodnotenie a uzatvorí element Zakon
  xwriter.WriteStartElement("Hodnotenie"); // <Hodnotenie
  xwriter.WriteAttributeString("Znamka", "0"); // Znamka="0"
  xwriter.WriteAttributeString("Pocet", "0");  // Pocet="0"
  xwriter.WriteAttributeString("Cas", teraz);  // Cas="teraz"
  xwriter.WriteEndElement(); // </Hodnotenie>
  xwriter.WriteEndElement(); // </Zakon>
}
```

XML súbor, ktorý bol vytvorený uvedeným kódom, v zobrazení Internet Explorerom zachytáva obr. 2.14.



Obr. 2.14. Výsledok transformácie textu do XML súboru.

Vytvorme obsluhu tlačidla Načítaj XML. Naším cieľom je zobraziť získaný XML súbor v data gride. Pre dosiahnutie tohto cieľa v súbore Form1.cs doplňme možnosť použitia priestoru mien

```
using System.Data;
```

Deklarujme členskú premennú dsMuza triedy Form1:

```
private DataSet dsMuza = new DataSet("MurphyhoZakony");
```

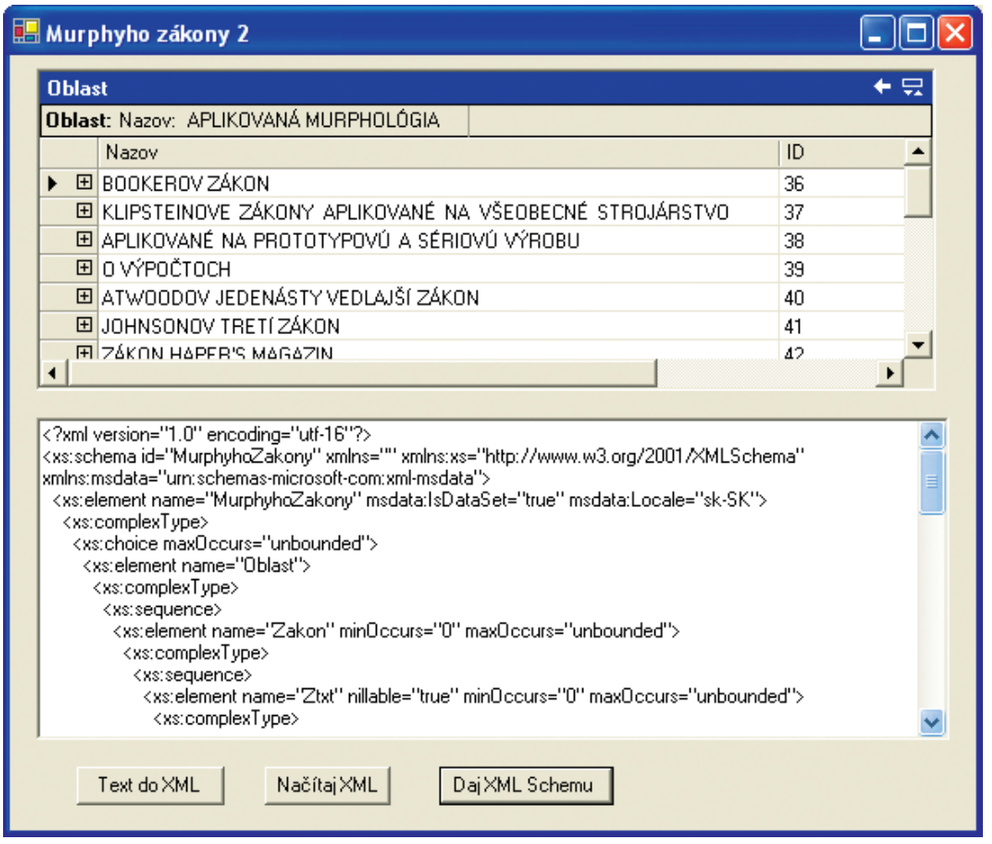
Definujme obsluhu tlačidla Načítaj XML:

```
private void ButNacitajXML_Click(object sender, System.EventArgs e)
{ // IB ************************
  // Do data setu dsMuza načíta XML súbor.
  dsMuza.ReadXml("MurphyhoZakony_02.xml");
  dataGrid1.DataSource = dsMuza;
  dataGrid1.DataMember = "Oblast";
  dataGrid1.CaptionText = dataGrid1.DataMember;
}
```

Objekt typu DataSet (v našom prípade členská premenná dsMuza) vie poskytnúť XML schému. Ukážeme to v obsluhe tlačidla Daj XML Schému:

```
private void ButDajXMLSchemu_Click(object sender, System.EventArgs e)
{ // IB *************************
  // Z data setu zistí XML schému a vypíše ju do text boxu.
  System.IO.StringWriter swXML = new System.IO.StringWriter();
  dsMuza.WriteXmlSchema(swXML);
  textBox1.Text = swXML.ToString();
}
```

Obsluha tlačidla Načítaj XML aj Daj XML Schému sú veľmi jednoduché, ale výsledný efekt je pozoruhodný. Zachytáva ho obr. 2.15.



Obr. 2.15. XML súbor v data gride a jeho XML schéma.

Ak zhodnotíme, čo sme v tejto kapitole dosiahli, musíme konštatovať, že príliš veľa energie sme venovali získaniu XML súboru. Ak ho už máme, potom práca s ním za podpory tried .NET Framework je veľmi jednoduchá. Pripomeňme, že vytvorenie XML súboru tak, ako sme ho predviedli vyššie, či už pomocou XML Notepadu resp. priamym využitím triedy XMLTextWriter je ojedinelou úlohou. Je to preto, že viac než vytváranie XML súborov sa pred programátorov kladú úlohy súvisiace so spracovaním existujúcich XML dokumentov. Práve takéto úlohy budeme riešiť v nasledujúcich kapitolách.

3. Použitie XML súboru

Motto: FINAGLEHO PRAVIDLO
Do problematiky vnikáš správne vtedy, keď jej zvonka - zvnútra porozumieš skôr, akoby si začal so štúdiom.

V predošlej kapitole sme vytvorili XML súbor s obsahom Murphyho zákonov. Ukážeme, niektoré možnosti využitia tohto súboru. Cesta, ktorou sa vydáme bude viesť cez ukážky zobrazenia obsahu súboru po jeho transformácii do HTML, cez použitie XML súboru vo windows aplikácii, vo webovej aplikácii a nakoniec aj vo webovej službe.

3.1. XSL transformácia

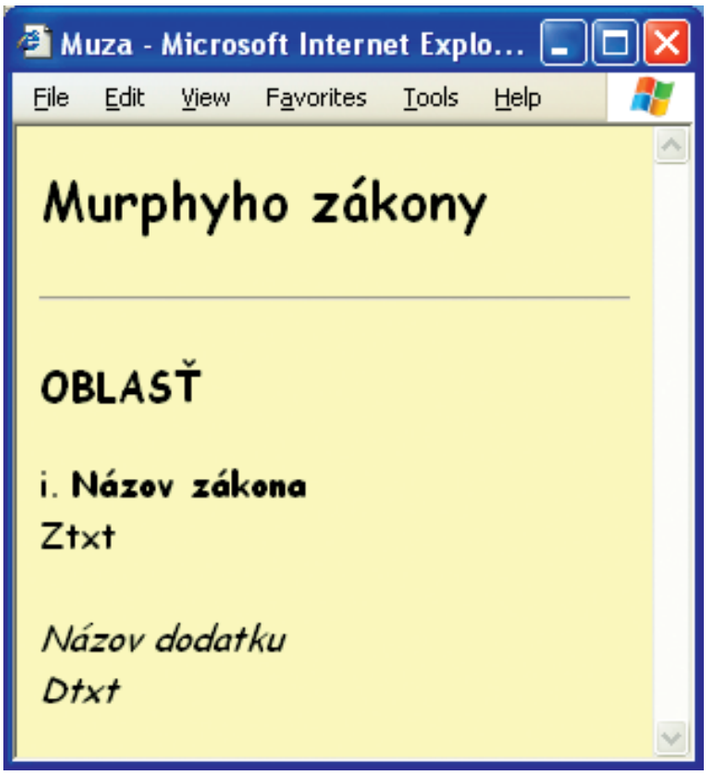
Jedným z čiastkových cieľov, ktoré má XML splniť v porovnaní s HTML, je dôsledné oddelenie obsahu dokumentu od jeho prezentácie. Skôr či neskôr sa dostaneme k potrebe vytvoriť prezentáciu obsahu XML dokumentu, alebo jeho časti. Popri možnosti určiť formu zobrazenia preto musí byť možnosť definovať predpisy pre výber určitých častí dokumentu, aj pre usporiadanie výslednej prezentácie údajov z dokumentu. Tie naviac nemusia zodpovedať postupnosti, v ktorej sú uvedené v XML dokumente. Tieto možnosti ponúka XSLT a v nej uplatnené zápisy vyhovujúce XPath.

Pod skratkou XSL sa skrýva označenie rozšíriteľného jazyka štýlov - Extensible Stylesheet Language. Konzorcium W3C zverejnilo odporúčanie definujúce tento jazyk v októbri 2001. XSLT (XSL Transformations) je časťou XSL. Tvorí jazyk pre definovanie transformácie XML dokumentu do iných XML dokumentov. Odporúčanie XSLT bolo zverejnené už v novembri 1999 V novembri 1999 bolo zverejnené aj odporúčanie XML Path Language (XPath). Je to jazyk, ktorý umožňuje zapísať výrazy adresujúce časti XML dokumentu. Tieto výrazy sa používajú v XSLT.

3.2. Transformácia XML súboru do HTML dokumentu

XSLT je jazyk určený pre transformáciu XML dokumentu do iného XML dokumentu. Dá sa však použiť aj na transformáciu XML dokumentu do HTML dokumentu. Postupujeme pri tom tak, že navrhneme „šablónu“ HTML dokumentu. Tá namiesto vlastných údajov dokumentu obsahuje predpisy, ako získať tieto údaje z XML dokumentu.

Pred vytvorením šablóny si treba urobiť predstavu - plán, akú podobu bude mať zobrazenie dokumentu. V prípade nášho XML dokumentu s obsahom Murphyho zákonov môžeme chcieť zobraziť všetky oblasti, všetky zákony s ich textami aj doplnkami.. Takáto predstava o výslednej podobe dokumentu sa dá vyjadriť v podobe obrázku. V našom prípade ju predstavuje obr. 3.1.



Obr. 3.1.: Predstava výslednej podoby dokumentu.

Predstava o výslednej podobe je realizovaná zápisom v HTML dokumente. Jeho kód je uvedený v tabuľke 3.1.

Tab. 3.1. Kód HTML dokumentu šablóny - predstavy podoby dokumentu.

```
 1 <HTML>
 2   <HEAD>
 3     <TITLE>Muza</TITLE>
 4   </HEAD>
 5   <BODY style="font-family: cursive ; background-color: #ffffcc;  text-color: #003300;">
 6     <H2>Murphyho zákony</H2>
 7     <!-- Pre každú oblasť -->
 8       <HR/>
 9       <H3>OBLASŤ</H3>
10       <!-- Pre každý zákon -->
11         <BR /> i. <B>Názov zákona</B>
12         <!-- Pre každý text zákona -->
13           <BR /> Ztxt
14         <!-- Koniec textov zákona -->
15         <!-- Pre každý dodatok -->
16           <I><BR /><BR />Názov dodatku
17           <!-- Pre každý text dodatku -->
18             <BR /> Dtxt
19           <!-- Koniec textov dodatku -->
20           </I>
21         <!-- Koniec dodatkov -->
22         <BR />
23       <!-- Koniec zákona -->
24     <!-- Koniec oblasti -->
25   </BODY>
26 </HTML>
```

V kóde HTML šablóny treba upozorniť na komentáre. Vyjadrujú, ktorých častí zdrojového XML dokumentu sa príslušná pasáž šablóny týka. Tak pasáž začínajúca komentárom **<!-- Pre každý zákon -->** (riadok 10) a končiaca komentárom **<!-- Koniec zákona -->** (riadok 23) vyjadruje, že v nej bude riešené zobrazenie týkajúce sa každého zákona. Ak skončíme zobrazenie údajov zákona (dosiahneme druhý z uvedených komentárov), treba sa vrátiť na miesto označené prvým komentárom. Je zrejmé, že sa tým vyjadruje potreba realizovať cyklus - opakovať zobrazenie údajov pre každý zákon. Popri komentároch sú v šablóne zvýraznené aj miesta, ktoré treba zaplniť údajmi z XML dokumentu. Je to OBLASŤ - názov oblasti, i - číslo zákona, Názov zákona, Ztxt, Názov dodatku a Dtxt.

Vytvorenú šablónu teraz ľahko môžeme pretvoriť na súbor, ktorý nazveme Muza1U.xsl. Číslo jedna v jeho názve vyjadruje, že je to naša prvá transformácia a písmeno U - že je použité kódovanie UTF-8.
Obsah tohto súboru je v tab. 3.2.

Tab. 3.2. Obsah súboru pre transformáciu XML dokumentu - Muza1U.xsl.

```
   <?xml version='1.0' encoding="utf-8"?>
   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:template match="/">
 1 <HTML>
 2   <HEAD>
 3     <TITLE>Muza</TITLE>
 4   </HEAD>
 5   <BODY style="font-family: cursive ; background-color: #ffffcc;  text-color: #003300;">
 6     <H2>Murphyho zákony</H2>
 7     <xsl:for-each select="MurphyhoZakony/Oblast"> <!-- Pre každú oblasť-->
 8       <HR />
 9       <H3><xsl:value-of select="@Nazov"/></H3>
10       <xsl:for-each select="Zakon"> <!-- Pre každý zákon -->
11         <BR /> <xsl:value-of select="@ID"/>. <B><xsl:value-of select="@Nazov"/></B>
12         <xsl:for-each select="Ztxt">  <!-- Pre každý text zákona -->
```

```
13          <BR /> <xsl:apply-templates />
14          </xsl:for-each>                    <!-- Koniec textov zákona -->
15          <xsl:for-each select="Dodatok"> <!-- Pre každý dodatok -->
16            <I><BR /><BR /><xsl:value-of select="@Nazov"/>
17            <xsl:for-each select="Dtxt">     <!-- Pre každý text dodatku -->
18              <BR /><xsl:apply-templates />
19            </xsl:for-each>                    <!-- Koniec textov dodatku -->
20            </I>
21            </xsl:for-each>                  <!-- Koniec dodatkov -->
22          <BR />
23        </xsl:for-each>                 <!-- Koniec zákona -->
24      </xsl:for-each>               <!-- Koniec oblasti -->
25    </BODY>
26  </HTML>
    </xsl:template>
    </xsl:stylesheet>
```

Porovnajme obsah tabuľky 3.1 a 3.2. Vidieť, že v transformačnom dokumente pribudli tri riadky na začiatku a dva na konci. Vytvorený predpis pre transformáciu XML dokumentu je tiež XML dokumentom. Má preto prológ XML dokumentu - prvý riadok. Druhý riadok je počiatočnou značkou prvku `xsl:stylesheet`. Obsahuje atribúty, ktorých údaje označujú, že budeme používať priestor mien xsl a akú verziu transformácie použijeme. Prvok `xsl:template` ohraničuje šablónu použitú pre výsledok transformácie. Posledné dva riadky v tab. 3.2. sú koncovými značkami uvedených prvkov. Ostatné riadky v tabuľke 3.2. majú vzor v tabuľke 3.1. Pre jednoduchšie porovnanie je v prvom stĺpci tabuliek uvedené číslovanie riadkov. Porovnávať treba riadky s rovnakým číslom.

Pred komentár `<!-- Pre každú oblasť -->` v riadku 7 tabuľky 3.1. je zaradené počiatočné ohraničenie prvku `xsl:for-each` v tabuľke 7.2.:

```
<xsl:for-each select="MurphyhoZakony/Oblast ">
```

Koncové ohraničenie tohto prvku je v riadku 24. Tento prvok bude obsahovať transformačné predpisy pre všetky prvky, ktoré vyhovujú výrazu zadanému v hodnote atribútu select. Zapisujú sa v jazyku XPath. Vyjadrujú hierarchiu prvkov transformovaného XML dokumentu. Veľmi to pripomína spôsob, akým sa určuje cesta k súborom v hierarchii adresárov (priečinkov) súborového systému. Stačí uviesť relatívnu cestu z adresára, v ktorom sa práve nachádzame. Zápisom „MurphyhoZakony/Oblast “ sa dostávame do prvku Oblast, ktorý je obsiahnutý v prvku MurphyhoZakony, pričom znak '/' slúži ako oddeľovač prvkov. Zápisom „MurphyhoZakony/Oblast /Zakon“ by sme sa dostali z vrcholu až do prvku Zakon.
Ak sme v prvku Oblast, výrazom „Zakon“ sa dostávame do prvku Zakon. Taká situácia je v riadku 10. Podobne sa dostávame do ďalších úrovní hierarchie prvkov: v riadku 12 sa dostávame do prvku Ztxt, v riadku 15 cez úroveň prvku Zakon do prvku Dodatok, v riadku 17 do prvku Dtxt.

Každý prvok `xsl:for-each` má svoje koncové ohraničenie. Tak riadok 24 je koncovým ohraničením prvku začínajúcom v riadku 7. Podobne sa dajú vidieť dvojice počiatočné - koncové ohraničenie v riadkoch 10 - 23, 12 - 14, 15 - 21 aj 17 - 19.

Prvky `xsl:value-of` slúžia na vloženie textu hodnoty atribútu. Tak zápis:

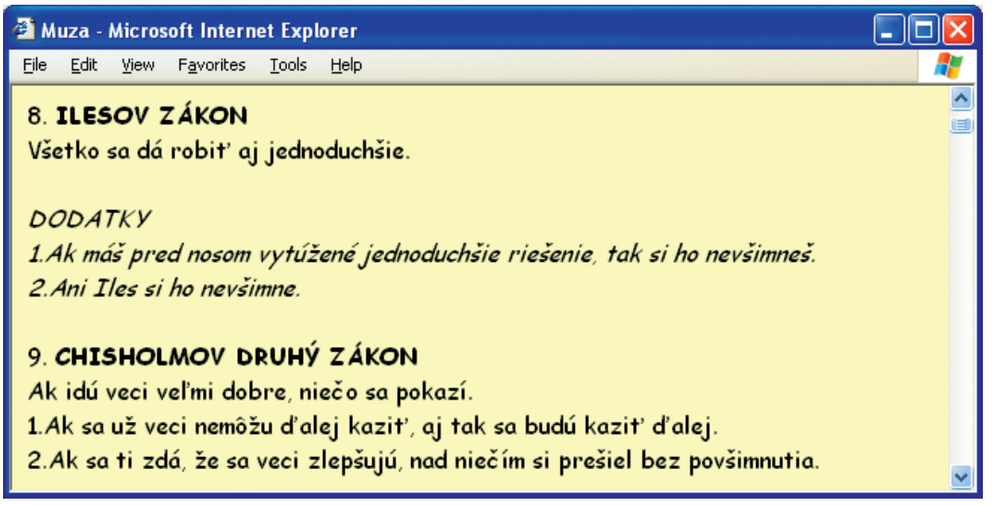
```
<xsl:value-of select="@Nazov"/>
```

v riadku 9 zabezpečí vloženie hodnoty atribútu Nazov prvku Oblast (znak @ vyjadruje, že sa jedná o atribút). Takým istým zápisom v riadku 11 je vložená hodnota atribútu Nazov prvku Zakon a v riadku 16 je vložený názov dodatku. V riadku 11 je vložená aj hodnota atribútu ID prvku Zakon. Vidieť teda, že vždy treba uvažovať aj s úrovňou - prvkom, v ktorom sa práve nachádzame.
Pre vloženie obsahu prvku Ztxt v riadku 13 je použitý zápis `<xsl:apply-templates />`. Podobne aj v riadku 18 sa tak vkladá obsah prvku Dtxt.

Uplatnenie opísaného súboru transformácie v XML dokumente MurphyhoZakony_02.xml dosiahneme tak, že za prológ XML dokumentu vložíme riadok:

```
<?XML:stylesheet type="text/xsl" href="Muza1U.xsl"?>
```

Po otvorení XML dokumentu internetovým prehliadačom získame zobrazenie jeho obsahu, ktoré ukazuje obr. 3.2.

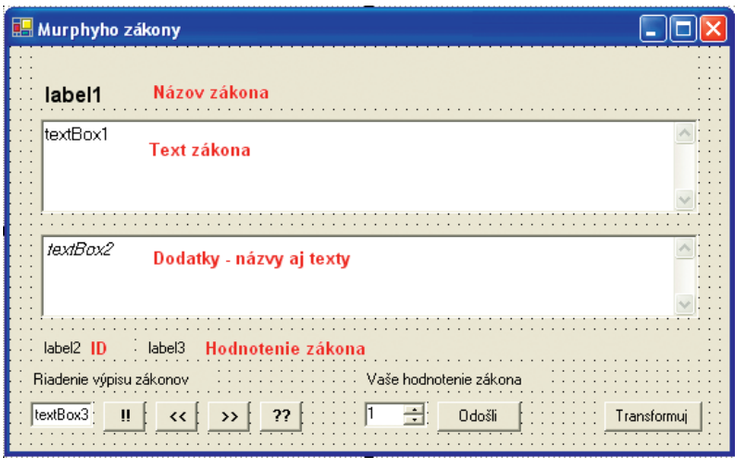


Obr. 3.2. Zobrazenie XML dokumentu s použitím XSL transformácie.

Ukázané konštrukcie v jazyku XSLT a v ňom využité možnosti ponúkané jazykom XPath dávajú rôznorodé možnosti manipulácie s obsahom XML dokumentu. Okrem použitých prvkov sa dajú využiť aj prvky `xsl:if`, `xsl:choose`, `xsl:when`, `xsl:otherwise` a ďalšie. Jazyky XSLT a XPath ponúkajú aj rôzne funkcie, napr. `current()`, `format_number()`, `count()`, `round()`, `sum()`.

3.3. Práca s XML súborom vo windows aplikácii

V predošlej kapitole sme po vytvorení XML súboru ukázali jeho zobrazenie s využitím objektov typu DataSet a DataGrid. Jednoduchosť takého zobrazenia je daná tým, že trieda DataSet má metódu ReadXml, ktorá všetko zariadi. „Keď je somárovi dobre, ide na ľad“, vraví sa v našich končinách. A tak aj my sa neuspokojíme s predošlým jednoduchým riešením, ale úlohu si sťažíme. Budeme chcieť manipulovať s obsahom XML súboru inými prostriedkami, než je DataSet a DataGrid.
V Microsoft Visual Studiu .Net vytvorme nový projekt typu C# so šablónou Windows Application. Nazvime ho napr. Muza3. V návrhovom zobrazení začleňme do formulára ovládacie prvky, ktoré ukazuje obr. 3.3.



Obr 3.3. Formulár aplikácie Muza3.

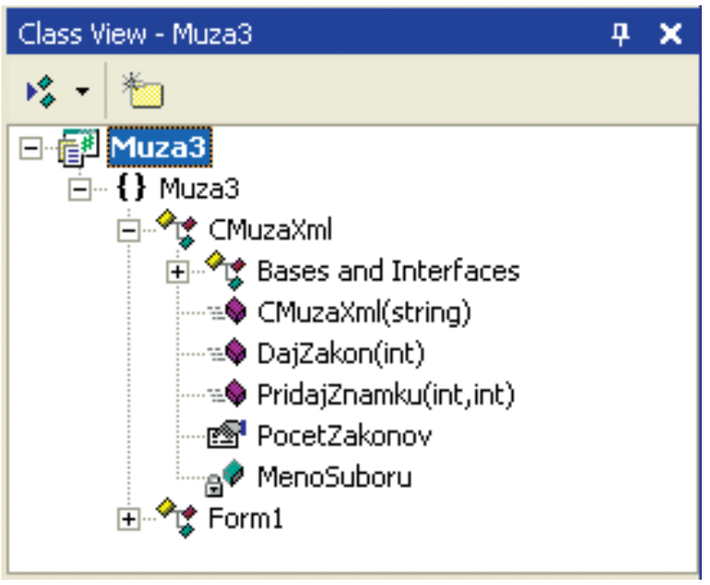
Budeme zobrazovať vždy iba údaje jedného Murphyho zákona, ktoré sú v obrázku uvedené červenou farbou. V ľavom dolnom rohu formulára sú ovládacie prvky pre riadenie výpisu zákonov. V zadávacom poli textBox3 je možné zadať identifikátor (poradové číslo) Murphyho zákona, ktorý potom môžeme vyvolať tlačidlom s dvoma výkričníkmi. Ďalšie dve tlačidlá slúžia pre zobrazenie predošlého a nasledujúceho zákona. Tlačidlom s dvoma otáznikmi sa vyvolá zobrazenie zákona, ktorého identifikačné číslo je odvodené od generátora náhodných čísel. Používateľ má možnosť zvoliť známku 1 až 5 pre zobrazený zákon a tlačidlom Odošli vyvolať pridanie zvolenej známky do hodnotenia zákona. V pravom dolnom rohu je tlačidlo Transformuj. V jeho obsluhe je ukázané, ako jednoducho sa dá programovo dosiahnuť transformácia XML súboru do HTML.

V opise programového riešenia postavenej úlohy začneme obsluhou tlačidla Transformuj. V súbore Form1.cs doplňme definíciu použitia priestorov mien:

```
using System.Xml;
using System.Xml.Xsl;
using System.Xml.XPath;
Obsluhu tlačidla Transformuj definujme takto:
private void ButTransform_Click(object sender, System.EventArgs e)
{ // IB **********************
  // Vytvorím XslTransform objekt.
  XslTransform xslt = new XslTransform();
  // Zavediem transformačný súbor.
  xslt.Load("Muza1U.xsl");
  // Urobím transformáciu XML súboru do HTML.
  xslt.Transform("MurphyhoZakony_02.xml", "Muza1.html");
  MessageBox.Show("Koniec transformácie");
}
```

Výsledkom bude súbor Muza1.html s Murphyho zákonmi. Ako zdroj je použitý súbor MurphyhoZakony_02.xml. Predpisy pre transformáciu sú čerpané zo súboru Muza1U.xsl. Určite nebudeme prekvapení, že po zobrazení získaného HTML súboru dostaneme Murphyho zákony vo forme, ktorú ukazuje obr. 3.2.

Pre vyriešenie zvyšnej časti postavenej úlohy vytvorme triedu CMuzaXml. Pohľad na túto triedu zachytáva obr. 3.4. Zdrojový text je v tab. 3.3.



Obr. 3.4. Trieda CMuzaXml.

Tab. 3.3. Zdrojový súbor triedy CMuzaXml.

```
using System;
using System.Xml;
using System.Xml.Xsl;
using System.Xml.XPath;
using System.Globalization;

namespace Muza3
{
  /// <summary>
  /// CMuzaXml je trieda, ktorá slúži pre prácu s XML súborom
  /// Murphyho zákonov.
  /// </summary>
  public class CMuzaXml
  {
    /// <summary>
    /// Meno súboru, v ktorom sú Murphyho zákony vo forme XML
    /// </summary>
    private string MenoSuboru;

    public CMuzaXml(string Subor)  { MenoSuboru=Subor; }

    /// <summary>
    /// Počet zákonov, ktoré sú zaznamenané v XML dokumente
    /// </summary>
    public int PocetZakonov
    { // IB ***************
      get
      { //Zistím počet zákonov v dokumente
        XmlDocument doc = new XmlDocument();
        doc.Load(MenoSuboru);
        XmlElement root = doc.DocumentElement;
        XmlNodeList nodeList = root.SelectNodes
          ("/MurphyhoZakony/Oblast/Zakon");
        return nodeList.Count;
      }
    }

    /// <summary>
    /// Metóda vráti textový reťazec zákona so zadaným ID vo forme XML
    /// </summary>
    public string DajZakon(int ID)
    { // IB **************
      XmlDocument doc = new XmlDocument();
      doc.Load(MenoSuboru);
      XmlElement root = doc.DocumentElement;
      string x="<Zakon Nazov='XXX CHYBA XXX' ID='0'>" +
          "<Ztxt>Nepodarilo sa získať zákon " +  ID.ToString()+
          "  ;-(</Ztxt></Zakon>";
      string Vyber="/MurphyhoZakony/Oblast/Zakon[@ID='" +
          ID.ToString() + "']";
      XmlNodeList ZoznamZakonov = root.SelectNodes(Vyber);
      if (ZoznamZakonov.Count>=1) {
        XmlNode Zakon = ZoznamZakonov.Item(0);
        x=Zakon.OuterXml;
      }
      return x;
    }

```

```
59      /// <summary>
60      /// Do hodnotenia zákona so zadaným ID pridá novú známku.
61      /// </summary>
62      public bool PridajZnamku(int ID, int NovaZnamka)
63      { // IB ****************
64        XmlDocument doc = new XmlDocument();
65        doc.Load(MenoSuboru);
66        XmlElement root = doc.DocumentElement;
67        string Vyber="/MurphyhoZakony/Oblast/Zakon[@ID='" +
68          ID.ToString()+"']/Hodnotenie";
69        XmlNodeList ZoznamHodnoteni = root.SelectNodes(Vyber);
70        if (ZoznamHodnoteni.Count>=1)
71        { XmlNode Hodnotenie = ZoznamHodnoteni.Item(0);
72          double Znamka = Double.Parse
73              (Hodnotenie.Attributes.Item(0).Value);
74          double Pocet = Double.Parse
75              (Hodnotenie.Attributes.Item(1).Value);
76          Znamka = ((double) NovaZnamka + Znamka*Pocet)/(Pocet+1);
77          Pocet+=1;
78          Hodnotenie.Attributes.Item(0).Value = Znamka.ToString("0.00");
79          Hodnotenie.Attributes.Item(1).Value = Pocet.ToString();
80          Hodnotenie.Attributes.Item(2).Value = DateTime.Now.ToString
81            ("s", DateTimeFormatInfo.InvariantInfo);
82          XmlTextWriter xwriter = new XmlTextWriter
83            (MenoSuboru, System.Text.Encoding.UTF8);
84          doc.Save(xwriter);
85          xwriter.Close();
86          return true;
87        }
88        return false;
89      }
90    }
91  }
```

Základom riešenia je trieda XmlDocument. Táto trieda v .NET Framework predstavuje základ implementácie odporúčaní W3C Document Object Model (DOM) Level 1 Core ako aj Core DOM Level 2. Tieto odporúčania definujú jazykovo nezávislé predpisy pre manipuláciu s XML dokumentom, ktorý je ako celok zavedený v pamäti. Umožňujú prístup k jednotlivým prvkom dokumentu, k ich atribútom aj obsahu. K tomu slúžia ďalšie triedy. V našom príklade sú použité XmlElement, XmlNodeList, XmlNode.

Pozrime sa na implementáciu vlastnosti (property) PocetZakonov (riadky 25 až 36). Vytvoríme objekt triedy XmlDocument (riadok 29). Použijeme metódu Load pre zavedenie XML súboru (jeho meno je v členskej premennej MenoSuboru, ktorá je nastavená v konštruktore). Potom získame koreňový element - objekt root typu XmlElement. Použijeme jeho metódu SelectNodes pre získanie objektu typu XmlNodeList. Parametrom je reťazec - výraz zapísaný v jazyku XPath. Definuje sa ním podmienka výberu. V našom prípade vyberáme všetky prvky Zakon (riadok 32 a 33). Získaný zoznam vo vlastnosti Count udržiava počet prvkov, ktoré vyhovujú zadanej podmienke.

Modifikáciou opísanej implementácie PocetZakonov je metóda DajZakon. Výraz XPath pre vytvorenie zoznamu je zapísaný v premennej Vyber (riadky 49, 50). V ňom definujeme aj požadovanú hodnotu atribútu ID prvku Zakon. Hľadaný Murphyho zákon je potom nultá položka získaného zoznamu (riadok 53), a tak vrátime jeho OuterXml. V prípade, že zadanej podmienke nevyhovuje ani jeden zákon (zle zadaná hodnota parametra ID), vo vrátenom reťazci je oznam o chybe.

Metóda PridajZnamku tiež začína zavedením XML súboru (riadok 65). V premennej Vyber definujeme podmienku pre výber zoznamu prvkov Hodnotenie, pričom atribút ID prvku Zakon musí mať zadanú hodnotu (riadky 67 a 68). Ak získame prvok Hodnotenie, vtedy najskôr zistíme hodnoty jeho atribútov Znamka a Pocet - nultá a prvá položka (riadky 72 až 75), potom urobíme výpočet novej známky a počtu hodnotení (riadky 76, 77) a zapíšeme nové hodnoty. Pre zápis zmeneného XML dokumentu vytvoríme objekt typu XmlTextWriter a dokumentu prikážeme, aby sa uložil (riadok 85).

Vytvorenú triedu využijeme vo formulári Form1 v metóde ZobrazZakon. Jej implementácia je uvedená v tab. 3.4.

Tab. 3.4. Implementácia metódy ZobrazZakon triedy Form1.

```
/// <summary>
/// Metóda ZobrazZakon pracuje s členskou premennou idZakona.
/// Je to číslo zákona, ktorý treba zobraziť.
/// Zákon vo forme XML získame ako textový reťazec volaním metódy
/// DajZakon objektu typu CMuzaXml. Tento reťazec zavedieme do
/// objektu typu XmlDocument. Z neho potom vyberieme údaje a zapíšeme
/// ich do textových polí formulára.
/// </summary>
private void ZobrazZakon()
{ // IB ****************
  // Do editačného poľa zapíšem hodnotu ID
  textBox3.Text=idZakona.ToString();
  CMuzaXml MuzaXml = new CMuzaXml(SuborXML);
  XmlDocument doc = new XmlDocument();
  doc.LoadXml(MuzaXml.DajZakon(idZakona));
  // Jediný prvok dokumentu je Zakon
  XmlElement Zakon = doc.DocumentElement;
  // Vyberiem hodnoty atribútov:
  label1.Text=Zakon.Attributes.Item(0).Value;        // Nazov
  label2.Text="ID="+Zakon.Attributes.Item(1).Value; // ID
  // Vymažem obsah textových polí - zákon, dodatok, hodnotenie:
  textBox1.Text="";
  textBox2.Text="";
  label3.Text ="";
  // Pripravím "enumerátor" pre pohyb v obsahu prvku Zakon
  IEnumerator ienum = Zakon.GetEnumerator();
  while (ienum.MoveNext())
  { // Zoberiem aktuálny prvok - node a spracujem ho
    XmlNode node = (XmlNode) ienum.Current;
    // Ak je to text zákona, pridám ho do prvého textového poľa
    if (node.Name=="Ztxt") textBox1.Text+=node.InnerText+"\r\n";
    if (node.Name=="Dodatok")
    { // Dodatok zapíšem do druhého textového poľa,
      // najskôr nadpis
      textBox2.Text=node.Attributes.Item(0).Value+"\r\n";
      // a potom všetky texty dodatku
      IEnumerator ienumDodatok = node.GetEnumerator();
      while (ienumDodatok.MoveNext())
      { XmlNode Dtxt = (XmlNode) ienumDodatok.Current;
        if (Dtxt.Name=="Dtxt") textBox2.Text+=Dtxt.InnerText+"\r\n";
      }
    }
    if (node.Name=="Hodnotenie")
    { // Hodnotenie je v troch atribútoch: Znamka, Pocet, Cas
      label3.Text ="Hodnotenie: známka="+node.Attributes.Item(0).Value;
      label3.Text+="  počet="+node.Attributes.Item(1).Value;
      label3.Text+="  čas "+node.Attributes.Item(2).Value;
    }
  }
  if (Zakon.Attributes.Item(1).Value=="0")
  { // Identifikátor získaného zákona je nula - je to chyba, oznámim
    // počet zákonov
    string sText = "Počet zákonov je "+MuzaXml.PocetZakonov.ToString();
    MessageBox.Show(sText);
  }
}
```

Podobne ako v triede CMuzaXml používame triedu XmlDocument z .NET Framework. Do objektu tejto triedy však nezavádzame celý XML súbor, ale iba textový reťazec. Využívame k tomu metódu LoadXml (riadok 15). Textový reťazec s obsahom zákona, ktorého číslo je v členskej premennej idZakona, získame z objektu typu CMuzaXml volaním metódy DajZakon. Jediným prvkom XML dokumentu, s ktorým budeme pracovať, preto bude Zakon. Získame ho v riadku 17. Z neho potom môžeme dostať hodnoty atribútu Nazov - nultá položka a ID - prvá položka (riadky 19 a 20). V obsahu prvku Zakon sú elementy Ztxt, Dodatok, v ňom Dtxt, a prvok Hodnotenie. Pre ich získanie vytvoríme objekt typu IEnumerator. Ten využijeme v cykle, v ktorom sa na ďalšie prvky dostávame metódou MoveNext (riadok 27). Objekt typu XmlNode získame od „enumerátora“ z jeho vlastnosti (property) Current (riadok 29). Objekt typu IEnumerator využijeme aj pre vyberanie textov dodatkov (riadky 37 až 41).

Vlastná obsluha tlačidiel pre riadenie výpisu Murphyho zákonov je po vytvorení metódy ZobrazZakon jednoduchá. Ukazuje to ich implementácia v tab. 3.5.

Tab. 3.5. Obsluha tlačidiel pre riadenie výpisu Murphyho zákonov v triede Form1.

```
 1 private void ButDajZakon_Click(object sender, System.EventArgs e)
 2 { // IB **********************
 3   idZakona=Int32.Parse(textBox3.Text);
 4   ZobrazZakon();
 5 }
 6 private void ButPredoslyZakon_Click(object sender, System.EventArgs e)
 7 { // IB ***************************
 8   if (idZakona>1) idZakona--;
 9   ZobrazZakon();
10 }
11 private void ButDalsiZakon_Click(object sender, System.EventArgs e)
12 { // IB ************************
13   idZakona++;
14   ZobrazZakon();
15 }
16 private void ButNahodnyZakon_Click(object sender, System.EventArgs e)
17 { // IB ***************************
18   CMuzaXml MuzaXml = new CMuzaXml(SuborXML);
19   int nZakonov = MuzaXml.PocetZakonov;
20   idZakona= GeneratorCisel.Next(1,nZakonov);
21   ZobrazZakon();
22 }
23
24 private void ButZnamka_Click(object sender, System.EventArgs e)
25 { // IB ********************
26   CMuzaXml MuzaXml = new CMuzaXml(SuborXML);
27   MuzaXml.PridajZnamku(idZakona, (int)numericUpDown1.Value);
28   ZobrazZakon();
29 }
```

Pre vytvorenie objektu typu CMuzaXml potrebujeme meno súboru. Je v členskej premennej SuborXML triedy Form1, ktorá je deklarovaná takto:

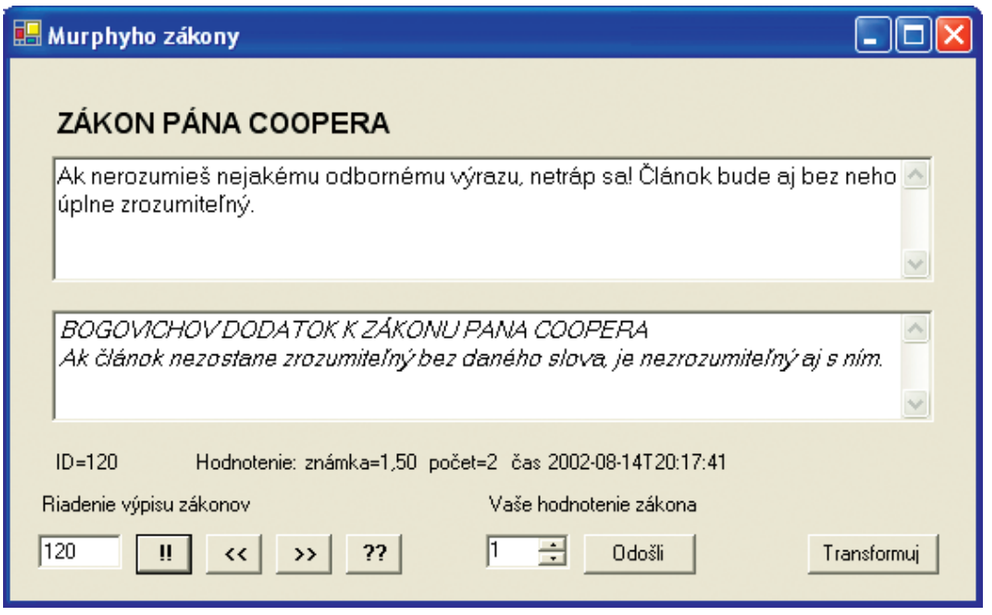
```
private string SuborXML="MurphyhoZakony_02.xml";
```

Za zmienku stojí aj členská premenná GeneratorCisel triedy Form1, ktorá je použitá v riadku 20. Je deklarovaná takto:

```
private System.Random GeneratorCisel = new System.Random();
```

V tabuľke 3.5. je uvedená aj obsluha tlačidla pre odoslanie hodnotenia. Zabezpečuje to metóda ButZnamka_Click (riadky 24 až 29).

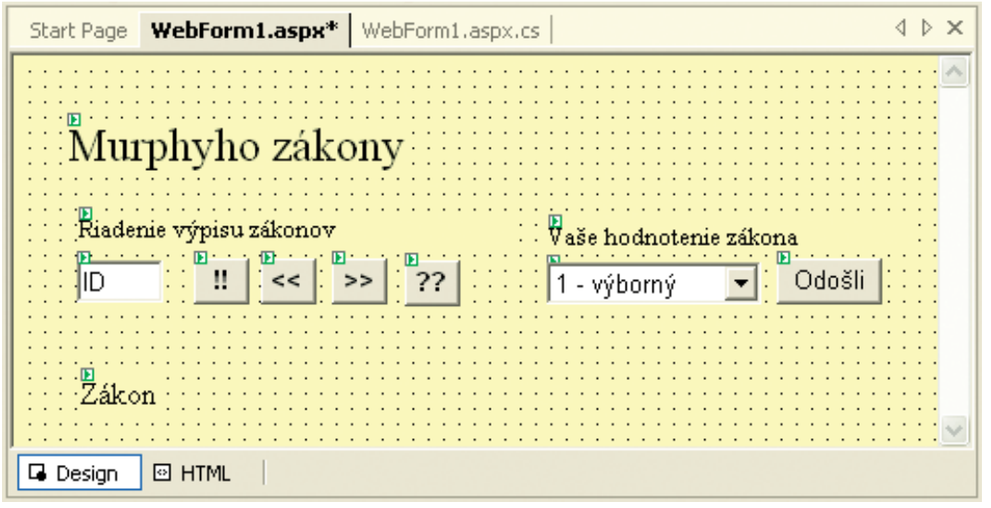
Okrem toho, že sme si pri tvorbe opisovanej aplikácie ukázali základné možnosti pre prácu s XML dokumentom, získali sme aj zaujímavú „hračku“. Dáva „náhode“ možnosť vybrať Murphyho zákon, ale používateľ si môže vybrať zákon aj „cielene“. Ukazuje to obr. 3.5.



Obr. 3.5. Aplikácia windows využívajúca XML súbor.

3.4. Práca s XML súborom vo webovej aplikácii

Vo Visual Studiu .NET založme nový projekt typu Visual C#. Použime šablónu ASP .NET Application. Zadajme umiestnenie aplikácie (Location) http://localhost/WAMuzaXml. Navrhnime formulár, ktorý ukazuje obr. 3.6.



Obr. 3.6. Formulár webovej aplikácie WAMuzaXml.

Uvedený obrázok naznačuje, že naším cieľom je vytvoriť webovú aplikáciu, ktorá bude ponúkať Murphyho zákony podobne, ako sme to ukázali v predošlej časti - v aplikácii windows. V danom čase budeme zobrazovať iba jeden zákon v spodnej časti formulára.

Uvedieme deklarácie, aby bolo vidieť, ako sú vytvorené ovládacie prvky pomenované:

```
protected System.Web.UI.WebControls.Label LabelNadpis;
protected System.Web.UI.WebControls.Label LabelRiadenie;
protected System.Web.UI.WebControls.TextBox IDZakona;
protected System.Web.UI.WebControls.Button ButDajZakon;
protected System.Web.UI.WebControls.Button ButPredosly;
protected System.Web.UI.WebControls.Button ButDalsi;
protected System.Web.UI.WebControls.Button ButNahodnyVyber;
protected System.Web.UI.WebControls.Label LabelHodnotenie;
protected System.Web.UI.WebControls.DropDownList DropDownZnamka;
protected System.Web.UI.WebControls.Button ButZnamka;
protected System.Web.UI.WebControls.Label LabelZakon;
```

Pre vyriešenie našej úlohy vytvoríme triedu CMuzaXml. Je to taká istá trieda, ako bola použitá v aplikácii windows - jej zdrojový kód je v tab. 3.4. Jediný rozdiel je v riadku 7, kde namiesto priestoru mien Muza3 je WAMuzaXml.

Podobne ako v riešení aplikácie windows, aj pri riešení webovej aplikácie vytvoríme metódy obsluhy udalostí tlačidiel. Aj tu najskôr vyriešime zobrazenie zákona, ktorý v podobe textu vo formáte XML získame z objektu triedy CMuzaXml. Implementácia metódy ZobrazZakon triedy WebForm1 je v tab. 3.6.

Tab. 3.6. Implementácia metódy ZobrazZakon triedy WebForm1.

```
 1 public void ZobrazZakon(int iz)
 2 { // IB ***************
 3   // Objekt typu CMuzaXml vie ponúknuť textový reťazec
 4   // s požadovaným zákonom vo formáte XML.
 5   CMuzaXml MuzaXml = new CMuzaXml(SuborXML);
 6   XmlDocument doc = new XmlDocument();
 7   doc.LoadXml(MuzaXml.DajZakon(iz));
 8   // Vytvorený XML dokument budeme transformovať.
 9   XslTransform xslt = new XslTransform();
10   // Zavedieme transformačný súbor.
11   xslt.Load(SuborXSL);
12   // Vytvoríme objekt, do ktorého bude zapísaný výsledok transformácie.
13   System.IO.StringWriter swriter = new System.IO.StringWriter();
14   xslt.Transform(doc,null,swriter);
15   // Vypíšeme výsledok transformácie.
16   LabelZakon.Text=swriter.ToString();
17   // Vypíšeme aj číslo zákona.
18   IDZakona.Text=iz.ToString();
19 }
```

V ponúknutom riešení je využitá možnosť XSL transformácie. Ňou získame výslednú podobu textu zákona. Je to modifikácia úlohy, ktorú sme riešili v časti 3.2 pri riešení transformácie XML do HTML súboru a tiež pri riešení obsluhy tlačidla Transformuj v časti 3.3. Potrebujeme k tomu objekt typu XslTransform. Metódou Load do neho zavedieme transformačný súbor (riadok 11). Pre vlastnú transformáciu použijeme metódu Transform. Ako parameter jej odovzdáme XML dokument, ktorý obsahuje požadovaný zákon. Výsledok transformácie je zapísaný do objektu typu StringWriter. Jeho text potom odovzdáme pre zobrazenie objektom LabelZakon (riadok 16).

Poznamenajme, že v členských premenných SuborXML a SuborXSL je meno XML súboru Murphyho zákonov, resp. meno s predpisom XSL transformácie. Ich deklarácia je takáto:

```
private string SuborXML;
private string SuborXSL;
Obsah týchto premenných je definovaný v metóde Page_Load triedy WebForm1 takto:
private void Page_Load(object sender, System.EventArgs e)
{
  SuborXML = MapPath("_vti_txt/MurphyhoZakony_02.xml");
  SuborXSL = MapPath("_vti_txt/Muza.xsl");
}
```

Vidieť z toho, že potrebné súbory sú v adresári _vti_txt aplikácie. Obsah transformačného súboru Muza.xsl je v tab. 3.7. Treba poznamenať, že súboru MurphyhoZakony_02.xml treba nastaviť vlastnosti umožňujúce jeho čítanie a zápis. Využijeme na to nástroje administrácie - Internet Information Services. Podobne umožníme čítať aj súbor Muza.xsl.

Tab. 3.7. Obsah transformačného súboru Muza.xsl.

```
<?xml version='1.0' encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="Zakon">
    <B><xsl:value-of select="@Nazov"/></B>
    <xsl:for-each select="Ztxt">  <!-- Pre každý text zákona -->
      <BR /> <xsl:apply-templates />
    </xsl:for-each>                  <!-- Koniec textov zákona -->
    <xsl:for-each select="Dodatok"> <!-- Pre každý dodatok -->
      <I>
      <BR /><BR /><B><xsl:value-of select="@Nazov"/></B>
     <xsl:for-each select="Dtxt">    <!-- Pre každy text dodatku -->
       <BR /><xsl:apply-templates />
    </xsl:for-each>                  <!-- Koniec textov dodatku -->
     </I>
    </xsl:for-each>                 <!-- Koniec dodatkov -->
    <BR />
    <xsl:for-each select="Hodnotenie">
      <BR /><B>Hodnotenie:</B>
      známka <B> <xsl:value-of select="@Znamka"/></B>
      počet <B> <xsl:value-of select="@Pocet"/></B>
    </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Obsluha tlačidiel WebForm1 je v tab. 3.8.

Tab. 3.8. Obsluha tlačidiel WebForm1.

```
private void ButDajZakon_Click(object sender, System.EventArgs e)
{ // IB *********************
  ZobrazZakon(Int32.Parse(IDZakona.Text));
}

private void ButPredosly_Click(object sender, System.EventArgs e)
{ // IB *********************
  ZobrazZakon(Int32.Parse(IDZakona.Text)-1);
}

private void ButDalsi_Click(object sender, System.EventArgs e)
{ // IB ******************
  ZobrazZakon(Int32.Parse(IDZakona.Text)+1);
}

private void ButNahodnyVyber_Click(object sender, System.EventArgs e)
{ // IB *************************
  CMuzaXml MuzaXml = new CMuzaXml(SuborXML);
  int nZakonov = MuzaXml.PocetZakonov;
  ZobrazZakon(GeneratorCisel.Next(1, nZakonov));
}

```

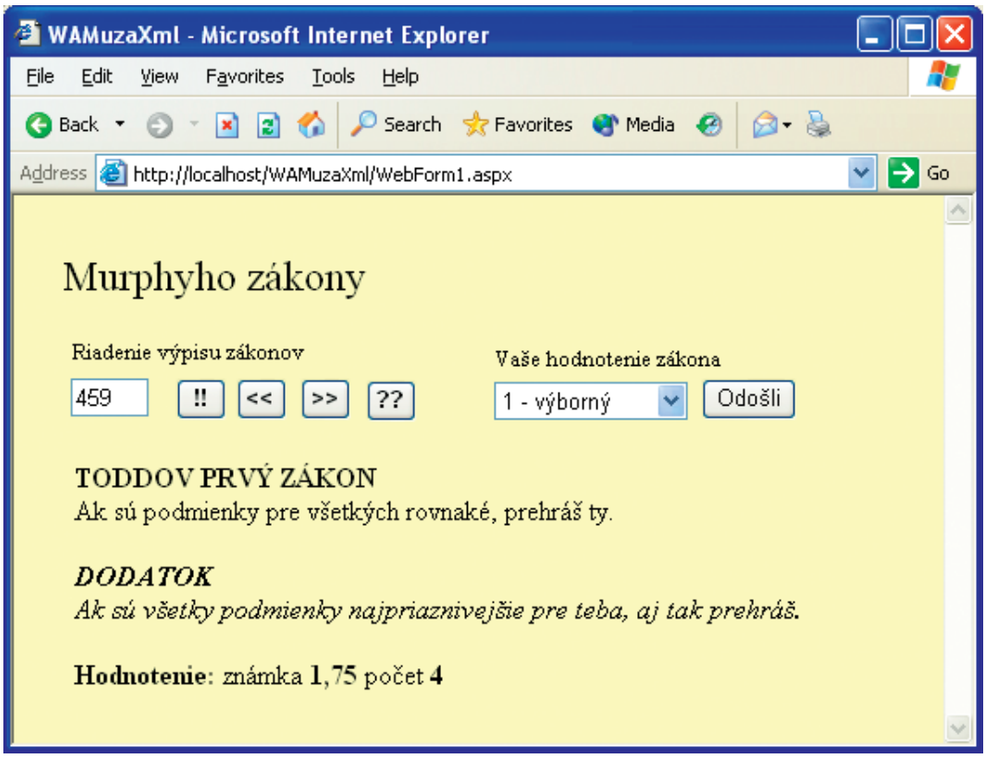
```
23 private void ButZnamka_Click(object sender, System.EventArgs e)
24 { // IB ********************
25   int iz = Int32.Parse(IDZakona.Text);
26   int Znamka = DropDownZnamka.SelectedIndex+1;
27   { CMuzaXml MuzaXml = new CMuzaXml(SuborXML);
28     MuzaXml.PridajZnamku(iz, Znamka);
29   }
30   ZobrazZakon(iz);
31 }
```

Využívame vyššie opísanú metódu ZobrazZakon. Základ pre číslo zákona, ktorý dávame zobraziť, čerpáme v štyroch prípadoch z textového poľa IDZakona (riadky 3, 8, 13, 25). V prípade obsluhy tlačidla pre náhodný výber zákona najskôr od objektu typu CMuzaXml zistíme počet zákonov a potom použijeme objekt GeneratorCisel (riadok 20). Je členskou premennou triedy WebForm1 deklarovanou takto:

```
private System.Random GeneratorCisel = new System.Random();
```

Pre úplnosť dodajme, že uvedené riešenie vyžaduje použitie priestorov mien:

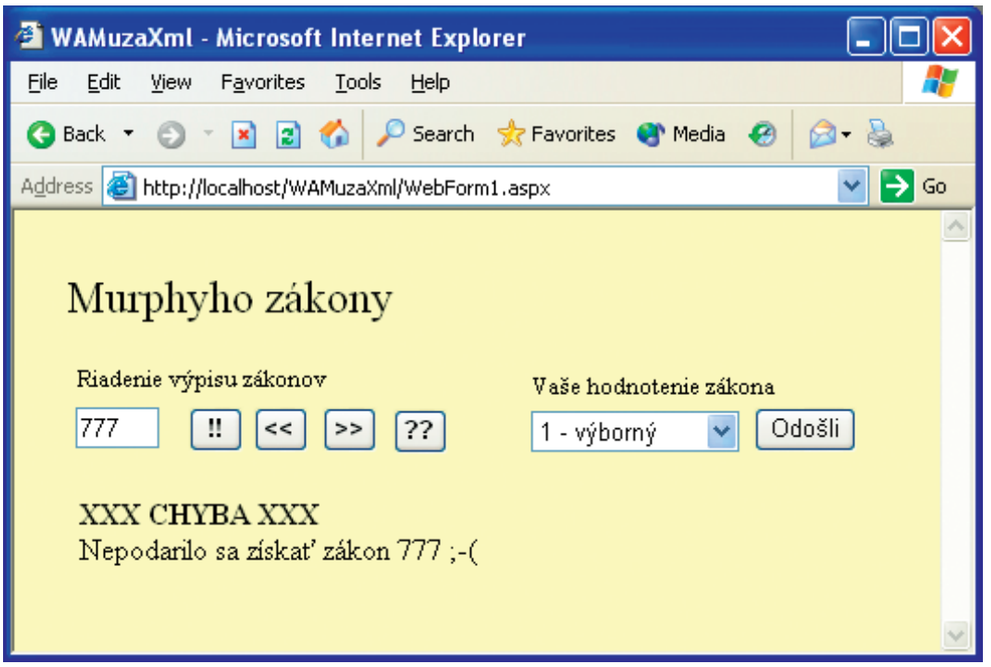
```
using System.Xml;
using System.Xml.Xsl;
using System.Xml.XPath;
```



Obr. 3.7. Výsledná podoba webovej aplikácie.

Výsledná podoba vytvorenej webovej aplikácie je na obr. 3.7.

Pre zaujímavosť ukážme, ako sa naša aplikácia zachová, ak používateľ zadá číslo zákona, ktoré presahuje počet zákonov v súbore (my ich máme 770 :-). Ukazuje to obr. 3.8.



Obr. 3.8. Oznam, že žiadaný zákon nie je.

Možno je namieste pripomenúť, že vytvorenú webovú aplikáciu je možné umiestniť na webový server. Vývojové prostredie MS Visual Studio .NET ponúka možnosť urobiť kópiu projektu. Stačí vyvolať menu Project - Copy project . K vytvorenej webovej aplikácii sa potom dá dostať internetovým prehliadačom.

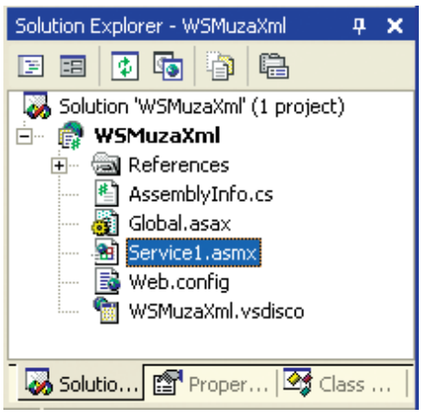
3.5. Práca s XML súborom vo webovej službe

Pojem webová služba je v súčasnosti veľmi často používaný. Oslovovaní sú v prvom rade manažéri. Ponúkajú sa im argumenty o tom, že vo webových službách môžu nájsť prostriedok pre vyriešenie mnohých problémov, ktoré súvisia s integráciou podnikových systémov. Veľké vyhliadky sa dávajú webovým službám v riešeniach B2C (business-to-consumer) aj B2B (business-to-business). Tvorbu vebových služieb umožňujú nové vývojové nástroje firmy Borland, IBM a samozrejme, aj firmy Microsoft.

Postavme si cieľ- vytvoriť možnosť práce s Murphyho zákonmi prostredníctvom webovej služby. Vytvorme možnosť získať počet zákonov. Ponúknime tiež možnosť získať obsah zákona so zadaným poradovým číslom. Umožnime tiež pridať do hodnotenia zákona zadanú známku. Môžeme počítať s tým, že Murphyho zákony sú uložené v súbore vo formáte XML.

Z uvedených požiadaviek vyplýva, že to, čo sme v predošlých dvoch častiach riešili v triede CMuzaXml, chceme teraz riešiť ako webovú službu.

Vo Visual Studiu .NET založme nový projekt typu Visual C#. Použime šablónu ASP .NET Service. Zadajme umiestnenie aplikácie (Location) http://localhost/WSMuzaXml . Chvíľu si budeme musieť počkať, kým vývojové prostredie nadviaže spojenie s definovaným webovým serverom - v našom prípade je to localhost. Znamená to, že budeme pracovať na lokálnom počítači a uvažujeme s IIS - Internet Information Server. V okne pohľadu na naše riešenie (solution) uvidíme, čo nám vývojové prostredie vytvorilo. Ukazuje to obr. 3.9.



Obr. 3.9. Pohľad na riešenie webovej služby.

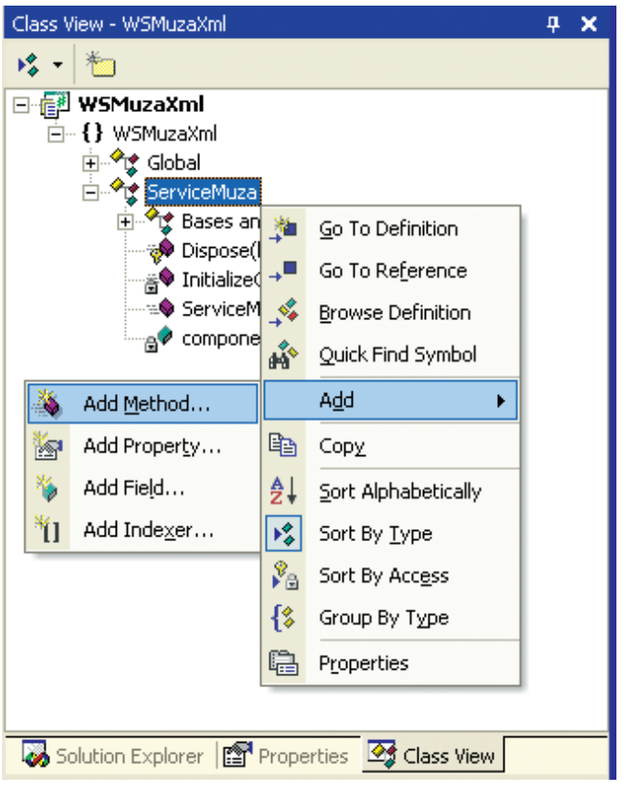
Sústredíme sa na službu, ktorá je pomenovaná ako Service1. S číslom 1, ktoré ponúklo vývojové prostredie, zrejme nebudeme spokojní. Premenujme našu službu na ServiceMuza. Ak si teraz zobrazíme zdrojový kód, uvidíme:

```
namespace WSMuzaXml
{
  /// <summary>
  /// Summary description for Service1.
  /// </summary>
  public class Service1 : System.Web.Services.WebService
  {
    public Service1()
    {
      //CODEGEN: This call is required by the ASP.NET Web Services Designer
      InitializeComponent();
    }
    // .. tu je v komentári príklad HelloWorld
  }
}
```

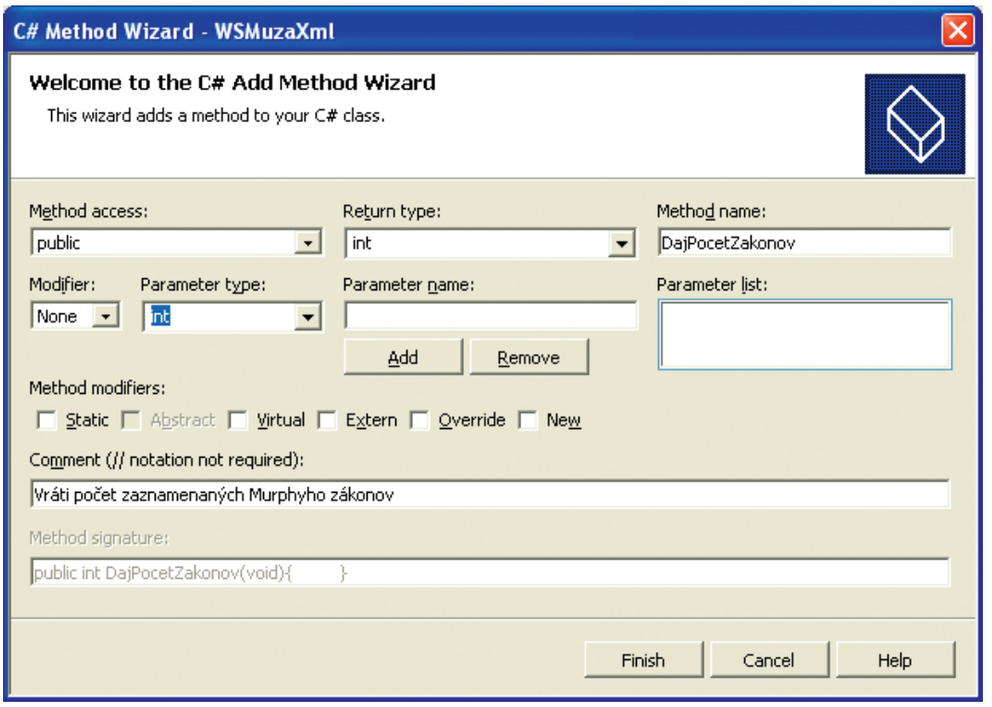
Aj tu nahraďme Service1 názvom ServiceMuza. K službe treba definovať priestor mien a rozumné je pridať aj jej opis. Pred definíciu triedy našej služby preto napíšme:

```
[WebService(Namespace = "http://buransky.sk/WSMuzaXml/",  Description =
"Ukážka využitia XML vo webových službách na príklade Murphyho zákonov.")]
```

Do triedy ServiceMuza pridajme novú metódu. Spôsob naznačuje obr. 3.10. Sprievodca pridaním novej metódy do triedy nám pomôže definovať potrebné údaje. Vidieť to na obr. 3.11. Zadali sme názov DajPocetZakonov a návratový typ int. Napísali sme aj text komentára.



Obr. 3.10. Pridanie metódy



Obr. 3.11. Sprievodca pridaním metódy.

V zdrojovom texte triedy ServiceMuza nájdeme kostru požadovanej metódy v tvare:

```
/// <summary>
/// Vráti počet zaznamenaných Murphyho zákonov
/// </summary>
public int DajPocetZakonov()
{
  return 0;
}
```

Doplňme pred definíciu tejto metódy, že je to metóda webovej služby a pridajme aj jej stručný opis:

```
[WebMethod(Description = "Vráti počet zaznamenaných Murphyho zákonov")]
```

Teraz už stačí doplniť kód, ktorým zistíme návratovú hodnotu.

Podobne budeme postupovať aj pri definovaní ďalších metód webovej služby. Vlastná implementácia webovej služby ServiceMuza je v tab. 3.9.

Tab. 3.9. Implementácia webovej služby ServiceMuza.

```
 1 using System;
 2 using System.Collections;
 3 using System.ComponentModel;
 4 using System.Data;
 5 using System.Diagnostics;
 6 using System.Web;
 7 using System.Web.Services;
 8 using System.Xml;
 9 using System.Globalization;
10
11 namespace WSMuzaXml
12 {
13   /// <summary>
14   /// ServiceMuza:
15   /// Ukážka využitia XML vo webových službách na príklade
16   /// Murphyho zákonov.
17   /// </summary>
18   [WebService(Namespace = "http://buransky.sk/WSMuzaXml/",
19    Description = "Ukážka využitia XML vo webových službách" +
20    "na príklade Murphyho zákonov.")]
21   public class ServiceMuza : System.Web.Services.WebService
22   {
23     /// <summary>
24     /// Meno súboru, v ktorom sú Murphyho zákony.
25     /// </summary>
26     private string MenoSuboru;
27     public ServiceMuza()
28     {
29       // CODEGEN: This call is required by the ASP.NET Web Services
30       // Designer
31    InitializeComponent();
32       // Nastavím meno súboru, v ktorom sú Murphyho zákony vo forme XML
33       MenoSuboru = Server.MapPath("_vti_txt/MurphyhoZakony_02.xml");
34     }
35
36     #region Component Designer generated code
37
38     //Required by the Web Services Designer
39     private IContainer components = null;
40
```

```
41      /// <summary>
42      /// Required method for Designer support - do not modify
43      /// the contents of this method with the code editor.
44      /// </summary>
45      private void InitializeComponent()
46      {
47      }
48
49      /// <summary>
50      /// Clean up any resources being used.
51      /// </summary>
51      protected override void Dispose( bool disposing )
53      {
54        if(disposing && components != null)
55        {
56          components.Dispose();
57        }
58        base.Dispose(disposing);
59      }
60
61      #endregion
62
63      /// <summary>
64      /// Vráti počet zaznamenaných Murphyho zákonov
65      /// </summary>
66      [WebMethod(
67        Description ="Vráti počet zaznamenaných Murphyho zákonov")]
68      public int DajPocetZakonov()
69      { // IB ******************
70        XmlDocument doc = new XmlDocument();
71        doc.Load(MenoSuboru);
72        XmlNodeList nodeList;
73        XmlElement root = doc.DocumentElement;
74        nodeList = root.SelectNodes("/MurphyhoZakony/Oblast/Zakon");
75
76        return nodeList.Count;
77      }
78
79      /// <summary>
80      /// Vráti Murphyho zákon so zadaným ID
81      /// </summary>
82      [WebMethod(Description = "Vráti Murphyho zákon so zadaným ID ")]
83      public string DajZakon(int ID)
84      { // IB **************
85        String x="<Zakon Nazov='XXX CHYBA XXX' ID='0'><Ztxt>Nepodarilo" +
86          " sa získať zákon " + ID.ToString() + " ;-(</Ztxt></Zakon>";
87        XmlDocument doc = new XmlDocument();
88        doc.Load(MenoSuboru);
89        XmlElement root = doc.DocumentElement;
90        String Vyber =
91          "/MurphyhoZakony/Oblast/Zakon[@ID='"+ID.ToString()+"']";
92        XmlNodeList ZoznamZakonov = root.SelectNodes(Vyber);
93        if (ZoznamZakonov.Count>=1)
94        {
95          XmlNode Zakon = ZoznamZakonov.Item(0);
96          x=Zakon.OuterXml;
97        }
98        return x;
99      }
100
```

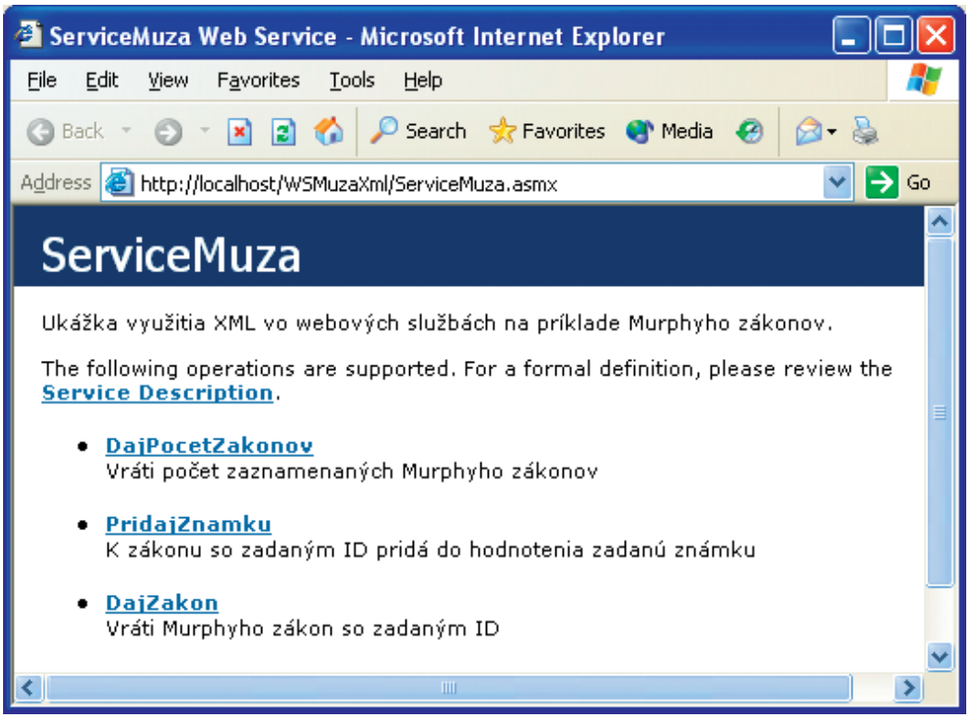
```
101     /// <summary>
102     /// K zákonu so zadaným ID pridá do hodnotenia zadanú známku
103     /// </summary>
104     [WebMethod(
105        Description = "K zákonu so zadaným ID pridá do hodnotenia" +
106         " zadanú známku")]
107     public int PridajZnamku(int ID, int Znamka)
108     {// IB ****************
109       XmlDocument doc = new XmlDocument();
110       doc.Load(MenoSuboru);
111       XmlElement root = doc.DocumentElement;
112       String Vyber = "/MurphyhoZakony/Oblast/Zakon[@ID='" +
113         ID.ToString() + "']/Hodnotenie";
114       XmlNodeList ZoznamHodnoteni = root.SelectNodes(Vyber);
115       if (ZoznamHodnoteni.Count>=1)
116       {
117         XmlNode Hodnotenie = ZoznamHodnoteni.Item(0);
118         double Znamka0 =
119           Double.Parse (Hodnotenie.Attributes.Item(0).Value);
120         double Pocet =
121           Double.Parse(Hodnotenie.Attributes.Item(1).Value);
122         Znamka0 = ((double) Znamka + Znamka0*Pocet)/(Pocet+1);
123         Pocet+=1;
124         Hodnotenie.Attributes.Item(0).Value = Znamka0.ToString("0.00");
125         Hodnotenie.Attributes.Item(1).Value = Pocet.ToString();
126         Hodnotenie.Attributes.Item(2).Value =
127           DateTime.Now.ToString("s", DateTimeFormatInfo.InvariantInfo);
128
129         XmlTextWriter xwriter = new
130            XmlTextWriter(MenoSuboru, System.Text.Encoding.UTF8);
131         xwriter.Formatting = Formatting.Indented;
132         doc.Save(xwriter);
133         xwriter.Close();
134         return ID;
135       }
136       return 0;
137     }
138
139     // WEB SERVICE EXAMPLE
140     // The HelloWorld() example service returns the string Hello World
141     // To build, uncomment the following lines then save and build the
142     // project.
143     // To test this web service, press F5
144
145 //  [WebMethod]
146 //  public string HelloWorld()
147 //  {
148 //    return "Hello World";
149 //  }
150   }
151 }
```

Uviedli sme celý obsah implementačného súboru webovej služby ServiceMuza z dvoch dôvodov. Prvým dôvodom je, aby sme dali možnosť porovnať riešenie implementácie webovej služby a samostatnej triedy - tab. 3.9 a tab. 3.3. Druhý dôvod je ukázať časti, ktoré vygenerovalo vývojové prostredie a časti napísané programátorom. Čísla riadkov, ktoré boli napísané programátorom, sú zvýraznené.

Z porovnania implementácie triedy a webovej služby môžeme urobiť záver, že sú tam iba drobné rozdiely. Pri projektovaní triedy je možné uvažovať s konštruktorom preberajúcim parametre. V triede CMuzaXml je to meno súboru s Murphyho zákonmi (tab. 3.3 riadok 20). Konštruktor triedy webovej služby je bez parametrov (tab. 3.9 riadok 27). V koštruktore webovej služby preto priamo určujeme hodnotu premennej MenoSuboru - tab. 3.9 riadok 33. V triede môžeme vytvoriť implementáciu vlastnosti - property. V našom prípade je to PocetZakonov v tab. 3.3. riadky 25 až 36. Vo webovej službe sme získanie počtu zákonov implementovali ako metódu DajPocetZakonov - tab. 3.9 riadky 66 až 77. Metódy, ktoré majú byť webovou službou zverejnené, musia mať definovaný atribút [WebMethod]. Potešiteľné je, že v triede aj vo webovej službe môžeme počítať s podporou tých istých tried .NET Framework. V našom prípade sú to triedy pre podporu práce s XML.

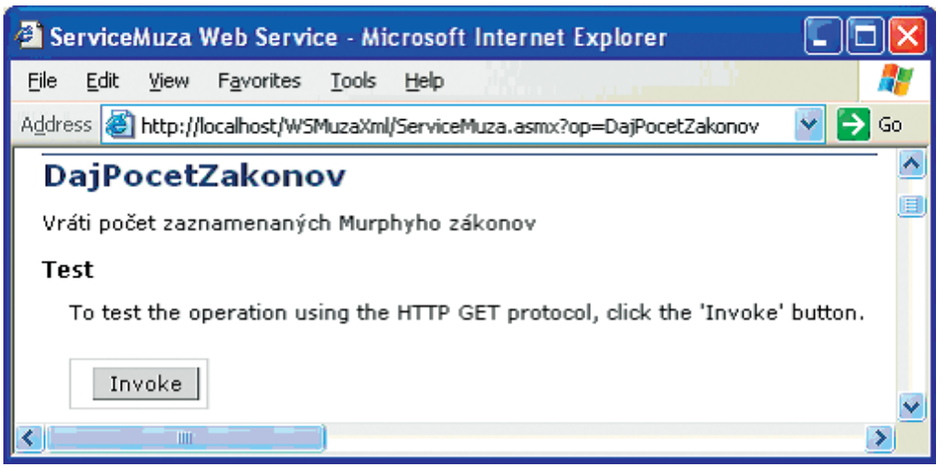
Ak sa pozrieme na tab. 3.9, môžeme vidieť, že vývojové prostredie vytvára kostru aplikácie. Sú tam dokonca oblasti, ktoré sa programátorovi neodporúčajú meniť. Programátorovi je vývojové prostredie všemožne nápomocné a vytvára mu podmienky, aby sa mohol sústrediť na riešenie podstaty webovej služby. Potešiteľný je aj príklad webovej metódy HelloWorld, ktorú vygenerovalo prostredie - je v spodnej časti tab. 3.9. Ak budete vytvárať prvú webovú službu, máte tam vzor, ako na to.

Nechajme vybudovať náš projekt s webovou službou a stlačme F5. Internet Explorer zobrazí charakteristiku vytvorenej webovej služby. Ukazuje to obr. 3.12.

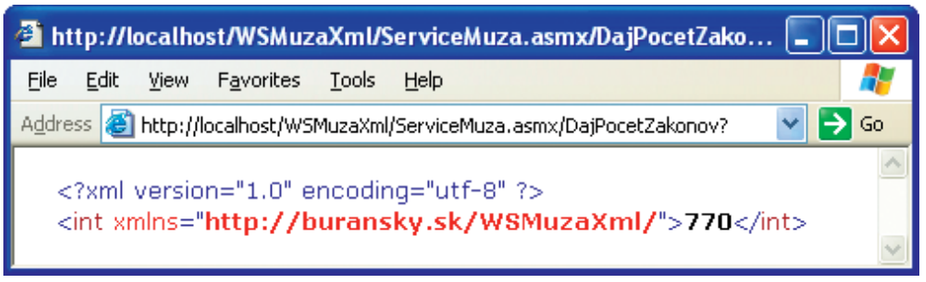


Obr. 3.12. Charakteristika webovej služby ServiceMuza.

Do adresára _vti_txt skopírujme súbor s Murphyho zákonmi a nastavme jeho vlastnosti tak, aby bolo povolené jeho čítanie aj zápis. A môžeme začať experimenty. Vyvolajme metódu DajPocetZakonov. Ukáže sa najskôr obr. 3.13 a po vyvolaní „Invoke“ obr. 3.14.

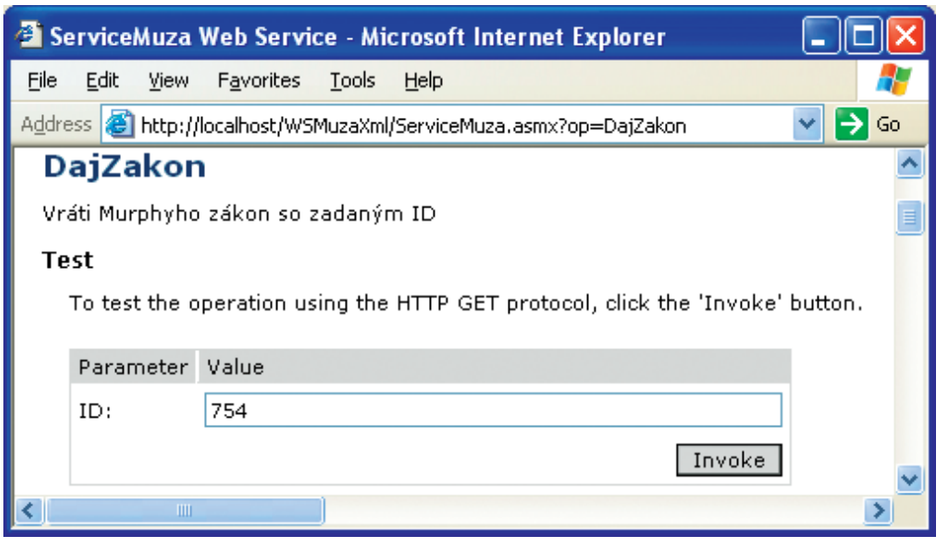


Obr. 3.13. Test metódy DajPocetZakonov.

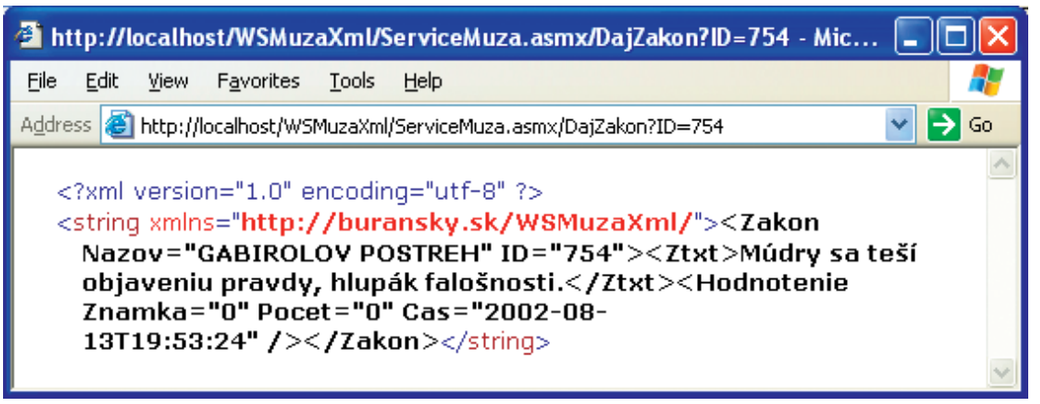


Obr. 3.14. Výsledok volania metódy DajPocetZakonov.

Podobne môžeme overiť aj metódu DajZakon. Ukazujú to obr. 3.15 a 3.16. V tomto prípade treba zadať parameter ID. Pri teste metódy PridajZnamku budeme musieť zadať dva parametre - identifikátor zákona aj známku.



Obr. 3.15. Test metódy DajZakon.

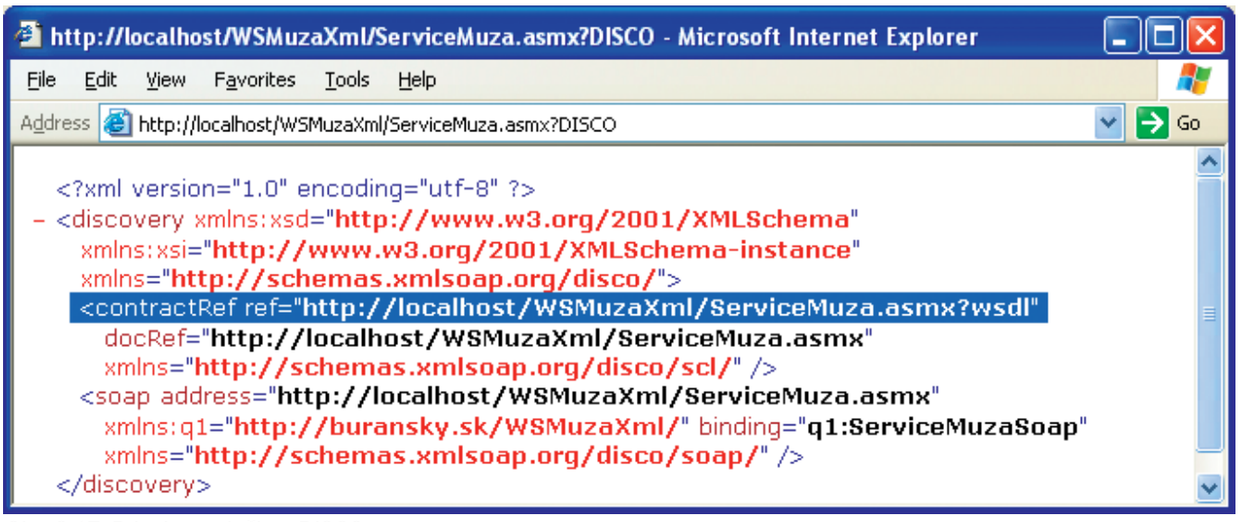


Obr. 3.16. Výsledok volania metódy DajZakon.

Ukázalo sa teda, že pracovať s XML súborom vo webovej službe nie je o nič náročnejšie, ako v aplikácii windows, či vo webovej aplikácii.

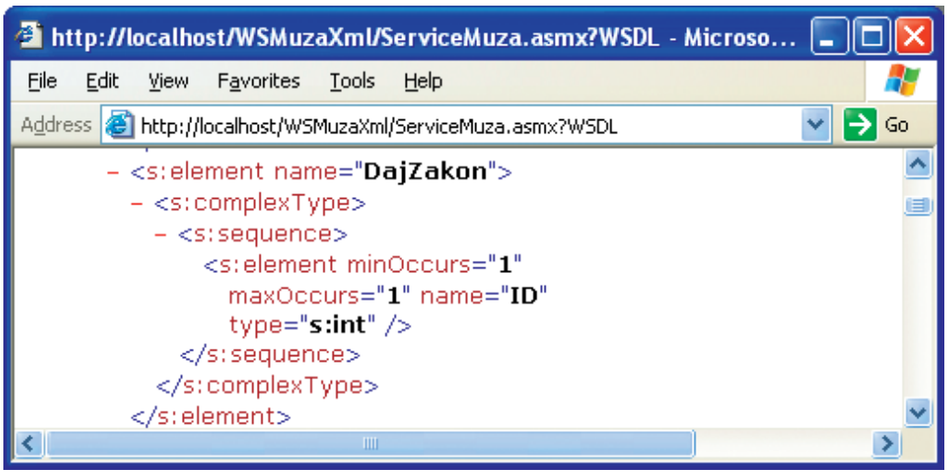
Vybudovanú webovú službu použijeme na vysvetlenie niektorých pojmov, ktoré s technológiou webových služieb súvisia.

Zadajme našej službe v http adrese parameter DISCO. Výsledok je na obr. 3.17.



Obr. 3.17. Prieskum služby - DISCO.

DISCO je technológia Microsoftu pre publikovanie a vyhľadávanie webových služieb. Ako ukazuje obr. 3.17, nami vytvorená webová služba po zadaní parametra DISCO podá základné informácie o tom, kde hľadať kontrakt s touto službou. Adresa (referencia) tohto kontraktu je na obr. 3.17 v zvýraznenom riadku. Vidieť, že ho získame, ak našej službe zadáme parameter WSDL. Poznamenajme, že ku kontraktu sa dostaneme aj zo zobrazenia charakteristiky webovej služby (obr. 3.12) vyvolaním Service Description - opis služby. Časť z neho vidieť na obr. 3.18.



Obr. 3.18. WSDL - kontrakt medzi serverom a klientom.

Skratka WSDL je odvodená od Web Services Description Language. Je to jazyk na opis webových služieb, založený na XML. Je definovaný v pripravovanom odporúčaní W3C. V júli 2002 mal štatút pracovnej skice (working draft in development). Na obr. 3.18 je fragment napísaný vo WSDL, ktorý je súčasťou opisu metódy DajZakon.

Uveďme ešte význam skratky SOAP - Simple Object Access Protocol. Je to protokol založený na XML. Určený je pre výmenu údajov v decentralizovanom distribuovanom prostredí. Pripravované je odporúčanie W3C. V júni 2002 bola SOAP verzia 1.2 v štádiu pracovných skicí (working drafts in last call). Má tri časti - Part 0: Primer, Part 1: Messaging Frame, Part 2: Adjuncts.

Nebudeme hlbšie prenikať do tajomstiev DISCO, UDDI (Universal Description, Discovery, and Integration), SOAP ani WSDL. Uspokojíme sa s tým, že existujú a že programátorom prinášajú pohodlie, ktorým sa použitie webovej služby veľmi približuje použitiu triedy. Webové služby sú budované preto, aby sa dali využiť v iných aplikáciách. Možno práve teraz, keď sme vytvorili webovú službu a videli jej prvé prejavy v ladiacom prostredí, je vhodné uviesť definíciu pojmu webová služba.

Webová služba je softvérová aplikácia identifikovaná prostredníctvom URI (Uniform Resource Identifier), ktorej interfejsy a väzby je možné definovať, opísať a vyhľadávať ako artefakty XML. Podporuje priamu interakciu s inými softvérovými aplikáciami prostredníctvom správ zapísanými v jazyku XML a prenášanými protokolmi internetu.

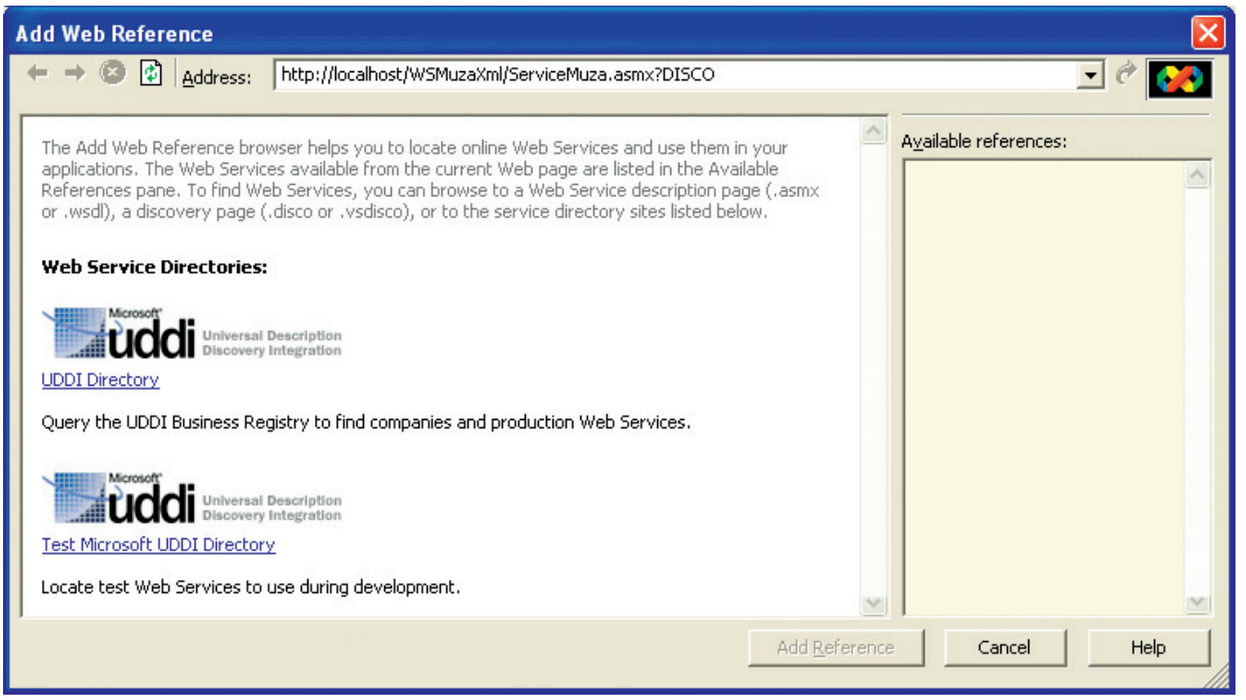
Ukážeme si použitie vytvorenej webovej služby vo webovej aplikácii.

3.6. Použitie webovej služby

Vo webovej aplikácii, ktorú sme riešili v projekte WAMuzaXml v časti 3.4, sme vytvorili triedu CMuzaXml. Umožňuje získať Murphyho zákon, počet zákonov a dokáže pridať známku do hodnotenia zákona. To isté poskytuje vo svojich metódach aj webová služba ServiceMuza, vytvorenie ktorej sme opísali v 3.5. Môžeme sa preto pokúsiť vo webovej aplikácii WAMuzaXml nahradiť triedu CMuzaXml webovou službou ServiceMuza.

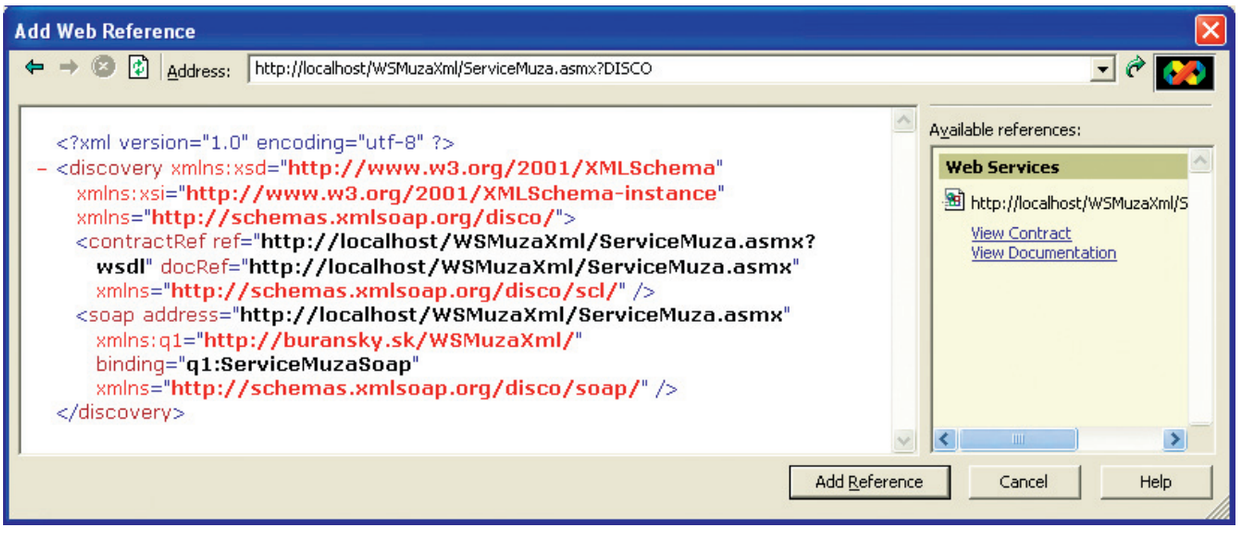
Otvorme projekt WAMuzaXml. Pridajme do projektu referenciu na webovú službu - menu Project - Add Web Reference. Vyvoláme tak sprievodcu - obr. 3.19. Zadajme adresu k našej webovej službe (na obr. 3.19 je už zadaná):

```
http://localhost/WSMuzaXml/WsMuzaXml.asmx?DISCO
```



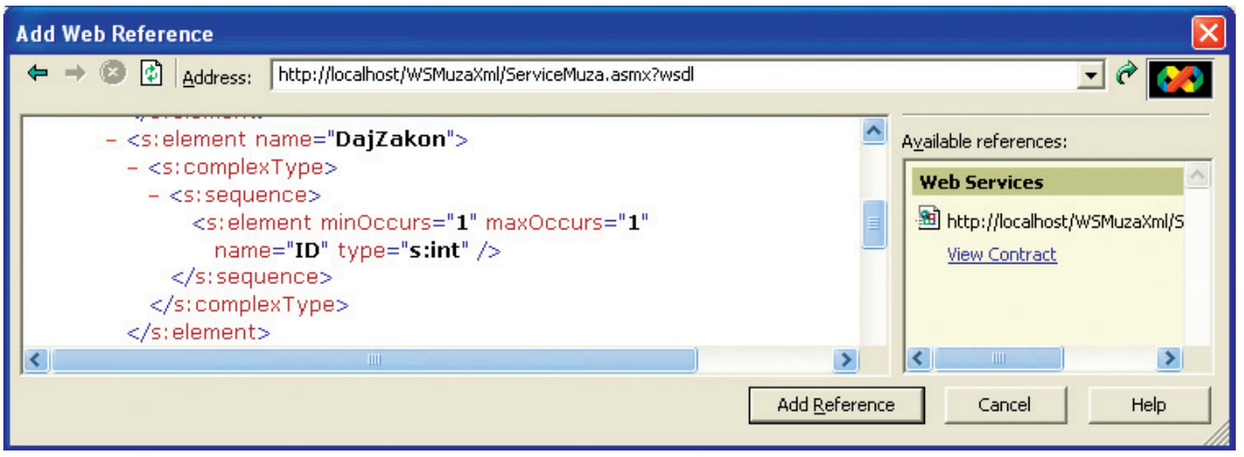
Obr. 3.19. Sprievodca pridania webovej referencie.

Po zadní uvedenej adresy prieskumník ukáže cestu ku kontraktu - obr. 3.20. Je to obdoba toho, čo sme skúmali v predošlej časti na obr. 3.17.



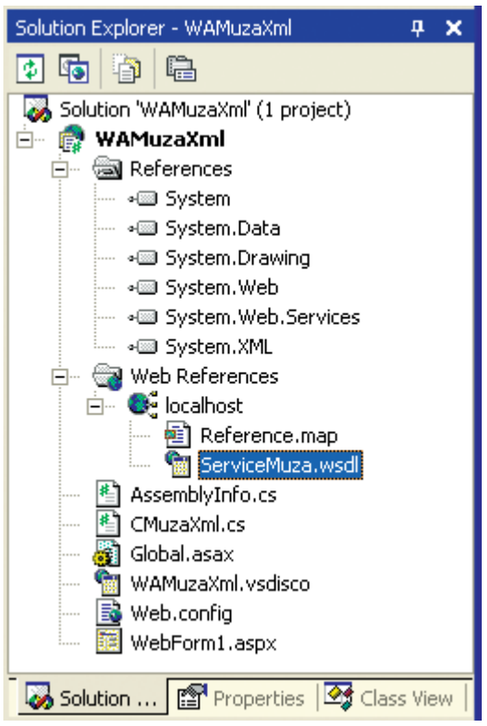
Obr. 3.20. Kontrakt s webovou službou.

Zvolme View Contract (link v pravej časti obr. 3.20). Dostaneme sa tak k WSDL opisu služby - obr. 3.21. Je zrejmé, že prvú fázu prieskumu môžeme vynechať, ak sprievodcovi pridávaním hneď na úvod v adrese namiesto parametra DISCO uvedieme parameter WSDL.



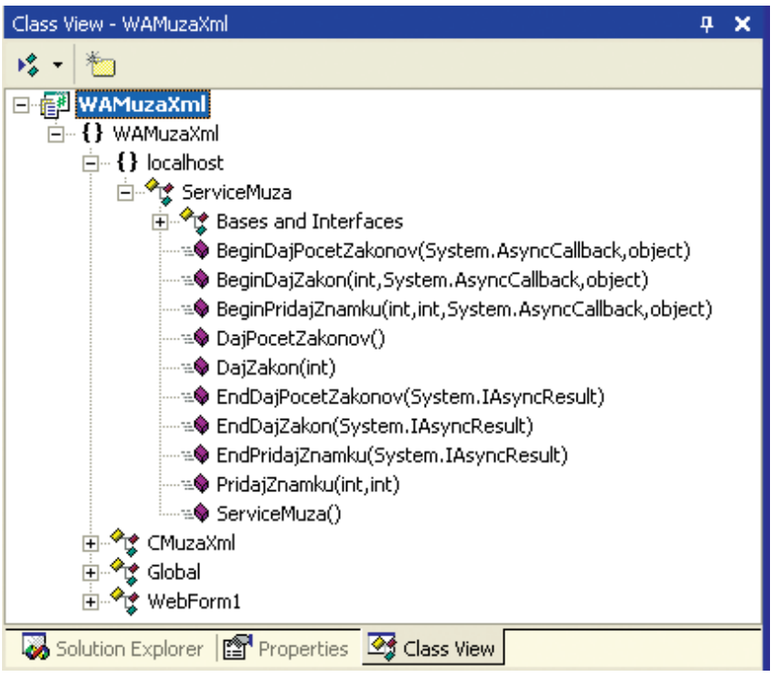
Obr. 3.21. WSDL opis služby.

Pridajme referenciu - tlačidlom Add Reference na obr. 3.21. Ak si potom pozrieme riešenie projektu, zistíme, že do neho pribudla referencia na webovú službu ServiceMuza - obr. 3.22.



Obr. 3.22. Webová služba v pohľade na riešenie.

V pohľade na triedy nájdeme triedu ServiceMuza - obr. 3.23.



Obr. 3.23. Trieda ServiceMuza.

Určite neunikne našej pozornosti, že trieda ServiceMuza má metódy DajPocetZakonov, DajZakon, PridajZnamku.

Upravme implementáciu triedy WebForm1. Pôvodné riešenie je v tab 3.6 a tab. 3.8. Upravené riešenie je v tab. 3.10.

Tab 3.10. Náhrada triedy CMuzaXml webovou službou ServiceMuza.

```
 1 public void ZobrazZakon(int iz)
 2 { // IB ***************
 3   // CMuzaXml MuzaXml = new CMuzaXml(SuborXML);
 4   ServiceMuza MuzaXml = new ServiceMuza();
 5   XmlDocument doc = new XmlDocument();
 6   doc.LoadXml(MuzaXml.DajZakon(iz));
 7   // ... pokračovanie ...
 8 }
 9 private void ButNahodnyVyber_Click(object sender, System.EventArgs e)
10 { // IB *************************
11   // CMuzaXml MuzaXml = new CMuzaXml(SuborXML);
12   // int nZakonov = MuzaXml.PocetZakonov;
13   ServiceMuza MuzaXml = new ServiceMuza();
14   int nZakonov = MuzaXml.DajPocetZakonov();
15   ZobrazZakon(GeneratorCisel.Next(1,nZakonov));
16 }
17 private void ButZnamka_Click(object sender, System.EventArgs e)
18 { // IB ********************
19   int iz = Int32.Parse(IDZakona.Text);
20   int Znamka = DropDownZnamka.SelectedIndex+1;
21   { // CMuzaXml MuzaXml = new CMuzaXml(SuborXML);
22     ServiceMuza MuzaXml = new ServiceMuza();
23     MuzaXml.PridajZnamku(iz, Znamka);
24   }
25   ZobrazZakon(iz);
26 }
```

Pôvodné riešenie je v tab. 3.10 v komentári. Nové riešenie znamená použitie ServiceMuza namiesto CMuzaXml. Čísla riadkov s novým riešením sú zvýraznené.

Do skúmania XML sme sa pustili preto, aby sme pochopili základy, na ktorých je postavená technológia webových služieb. Došli sme však k zaujímavému poznaniu. Pri tvorbe aplikácie, ktorá využíva webové služby, nepotrebujeme poznať XML. To je skryté niekde v pozadí. Prevažná väčšina programátorov v súčasnosti nepoužíva strojové inštrukcie procesora, ale využíva vymoženosti vyšších programovacích jazykov, knižnice tried, komponenty... Nikto však nepochybuje o tom, že v konečnom dôsledku všetko zabezpečia strojové inštrukcie. S XML to bude podobne. Bude v pozadí tvoriť základ pre technológie, také ako je WSDL, SOAP, UDDI. Vytvorenie zložitej aplikácie bude možné aj bez poznania detailov týchto technológií.

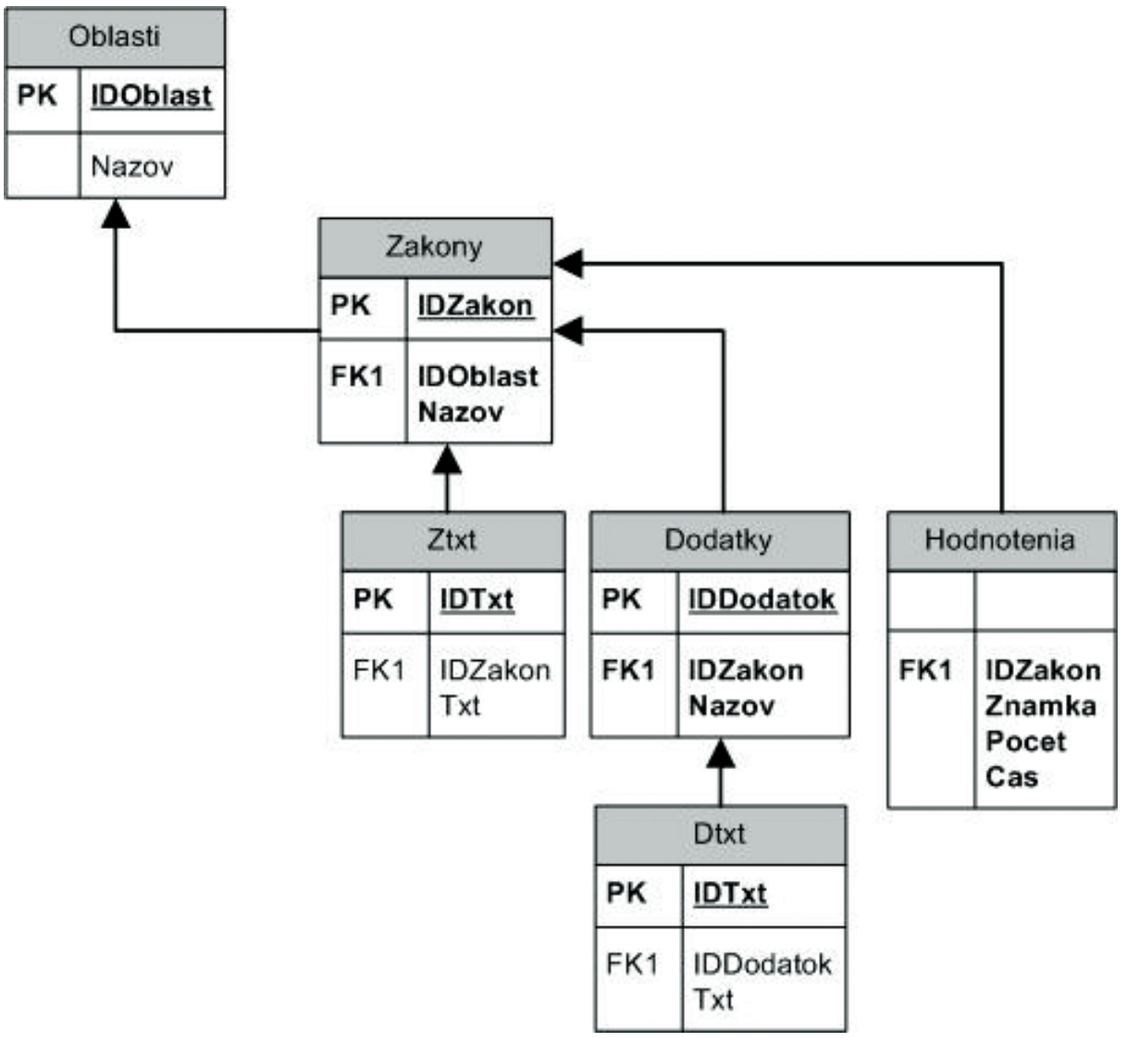
Pozrime sa na webové služby z pohľadu programátorov. Snaha zvýšiť efektívnosť ich práce vyústila v 80-tych rokoch minulého storočia do objektovo orientovaného programovania. Podstatou práce programátorov sa stalo objavenie objektov, tvorba a využívanie tried. Prekročiť **hranice jednej aplikácie,** t.j. opakované použitie vytvorených tried, sa najskôr dosahovalo pomocou knižníc tried, neskôr dynamicky linkovanými knižnicami. V prvej polovici 90-tych rokov to vyústilo do tvorby a využívania objektov COM (Component Object Model). V druhej polovici 90-tych rokov sa v podobe DCOM (distributed COM) podarilo prekročiť **hranice jedného stroja.** Bolo umožnené, aby v sieťovom prostredí program bežiaci na jednom stroji využíval triedy sídliace na inom stroji. Webové služby sú pokračovaním „rozpínania“ - umožňujú prekročiť **hranice jednej platformy.** Pri dodržaní štandardov pre opis webových služieb (WSDL), protokolu vzájomnej komunikácie objektov (SOAP), založených na textovom formáte XML, je umožnené použitie internetových protokolov pre vzájomnú spoluprácu programov bežiacich nielen na rôznych strojoch, ale dokonca aj na rôznych platformách. Aplikácie, využívajúce webové služby, môžu byť nielen iné webové služby, ale aj webové či klientské aplikácie. Na strane klientov môžu byť štandardné osobné počítače, ale aj zariadenia typu PDA (personal digital assistants), či mobilné telefóny. Pritom pre programátora je spôsob využívania webových služieb veľmi blízky spôsobu využívania tried.

4. SQL server a XML

Motto: PEEROV ZÁKON
Vyriešenie problému pozmení podstatu problému.

XML je ako vírus. Zasiahne, čo sa len dá. Zasiahol aj databázové stroje. XML „nákaza“ však nie je nedostatkom, skôr naopak - prednosťou. Sme svedkami toho, ako dodávatelia databázových strojov súťažia už nielen v tom, kto poskytne vyšší výkon, ale aj v tom, kto poskytne „lepšiu“ podporu XML.

Databázové stroje umožňujú vytvárať úložiská dát. XML je základom pre prenos dát medzi počítačmi. Je preto prirodzené žiadať, aby databázové stroje dokázali vydať údaje priamo vo formáte XML. Ak to dokážu, budeme uchránení od potreby robiť podobné programy, ako sme ukázali v druhej kapitole - vytvárať si „svoje vlastné“ XML súbory. Namiesto toho dáme príkaz databázovému stroju a ten nám vydá údaje priamo vo formáte XML aj z „klasických“ relačných databáz. Microsof SQL server to umožňuje tak, že rozšíril príkaz SELECT jazyka SQL o zápis predpisu, ktorý špecifikuje formu výstupu. Slúži k tomu klauzula FOR XML xxx, kde xxx bližšie určuje, akú formu XML žiadame.



Obr. 4.1. Štruktúra databázy Muza - Murphyho zákony.

Doteraz sme prenikali do XML prostredníctvom Murphyho zákonov. Budeme tak robiť naďalej. Využijeme k tomu databázu Muza spravovanú Microsoft SQL serverom 2000. Štruktúra tabuliek databázy je na obr. 4.1.

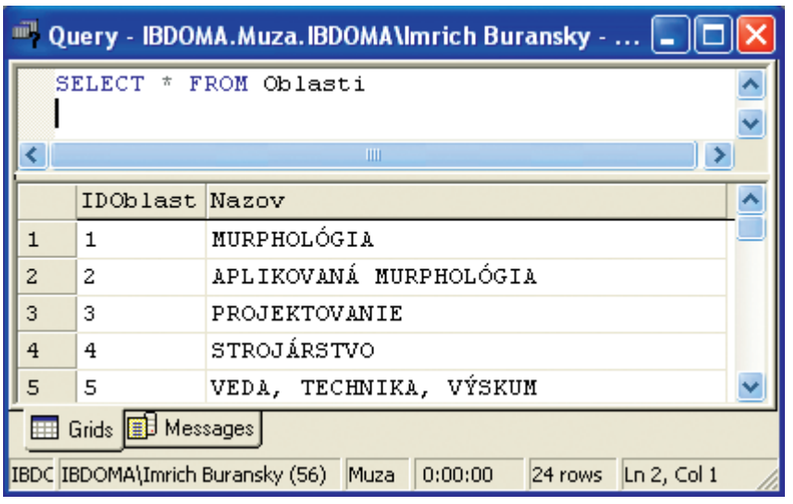
Vidíme, že sú tam oblasti, zákony so svojimi textami, dodatkami aj hodnotením.

4.1. Výber údajov z databázy vo formáte XML

Určite nás neprekvapí výsledok, ktorý ukáže SQL Query Analyzer po zadní príkazu:

```
SELECT * FROM Oblasti
```

Pozrime si ho na obr. 4.2.

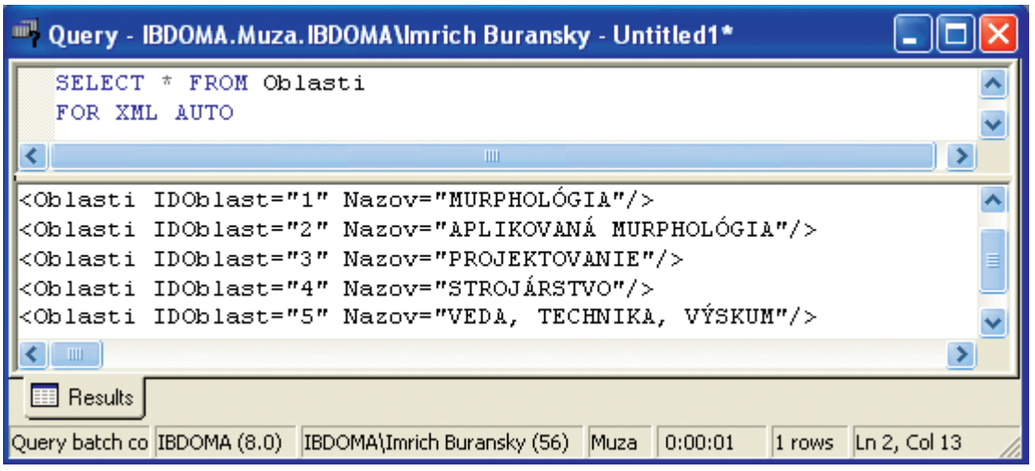


Obr. 4.2. Údaje z tabuľky Oblasti

Požiadajme, aby nám SQL server vydal údaje vo formáte XML. Dosiahneme to príkazom:

```
SELECT * FROM Oblasti
FOR XML AUTO
```

V príkaze `SELECT` je klauzula `FOR XML`. Slovo `AUTO` definuje mód - žiadanú formu výsledku. SQL Query Analyzer nám dá výsledok, ktorý je na obr. 4.3.



Obr. 4.3. Údaje z tabuľky Oblasti vo forme XML.

Každý riadok výsledku je v jednom XML elemente. Meno elementu je meno tabuľky - v našom príklade Oblasti. Mená stĺpcov sú atribútmi - IDOblast, Nazov. V hodnotách atribútov sú vlastné údaje z databázy.

Základná syntax klauzuly FOR XML je takáto:

```
FOR XML mode [, XMLDATA] [, ELEMENTS][, BINARY BASE64]
```

Argumenty majú takýto význam:

- **XML mode** .. určuje tvar výsledku, pričom mode môže byť RAW, AUTO, EXPLICIT.
- **XMLDATA** .. určuje, že súčasťou dokumentu bude aj XML schéma.
- **ELEMENTS** .. v móde AUTO sa určuje, že údaje budú dávané ako elementy. Inak sú údaje v hodnotách atribútov.
- **BINARY BASE64** .. určuje, že binárne dáta budú reprezentované vo formáte base64-encoded. Táto voľba musí byť určená pri získavaní údajov v móde RAW a EXPLICIT. V móde AUTO sú bez udania tejto voľby vracané ako referencia.

Urobme ešte zopár pokusov, aby sme ukázali, čo klauzula FOR XML dokáže. Budeme vyberať údaje z tabuľky Oblasti príkazom:

```
SELECT * FROM Oblasti WHERE IDOblast < 3
FOR XML xxx
```

kde na mieste xxx budeme meniť hodnoty argumentov. Výsledky sú zhrnuté v tab. 4.1.

Tab. 4.1. Možnosti klauzuly FOR XML - výber z tabuľky Oblasti.

```
SELECT * FROM Oblasti WHERE IDOblast < 3
FOR XML AUTO

<Oblasti IDOblast="1" Nazov="MURPHOLÓGIA"/>
<Oblasti IDOblast="2" Nazov="APLIKOVANÁ MURPHOLÓGIA"/>
```

```
SELECT * FROM Oblasti WHERE IDOblast < 3
FOR XML RAW

<row IDOblast="1" Nazov="MURPHOLÓGIA"/>
<row IDOblast="2" Nazov="APLIKOVANÁ MURPHOLÓGIA"/>
```

```
SELECT * FROM Oblasti WHERE IDOblast < 3
FOR XML AUTO, ELEMENTS

<Oblasti>
  <IDOblast>1</IDOblast>
  <Nazov>MURPHOLÓGIA</Nazov>
</Oblasti>
<Oblasti>
  <IDOblast>2</IDOblast>
  <Nazov>APLIKOVANÁ MURPHOLÓGIA</Nazov>
</Oblasti>
```

```
SELECT * FROM Oblasti WHERE IDOblast < 3
FOR XML AUTO, XMLDATA

<Schema name="Schema2" xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="Oblasti" content="empty" model="closed">
    <AttributeType name="IDOblast" dt:type="i4"/>
    <AttributeType name="Nazov" dt:type="string"/>
    <attribute type="IDOblast"/><attribute type="Nazov"/>
  </ElementType>
</Schema>
<Oblasti xmlns="x-schema:#Schema2" IDOblast="1" Nazov="MURPHOLÓGIA"/>
<Oblasti xmlns="x-schema:#Schema2" IDOblast="2" Nazov="APLIKOVANÁ MURPHOLÓGIA"/>
```

```
SELECT * FROM Oblasti WHERE IDOblast < 3
FOR XML AUTO, ELEMENTS, XMLDATA

<Schema name="Schema3" xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="Oblasti" content="eltOnly" model="closed" order="many">
    <element type="IDOblast"/>
    <element type="Nazov"/>
  </ElementType>
  <ElementType name="IDOblast" content="textOnly" model="closed" dt:type="i4"/>
  <ElementType name="Nazov" content="textOnly" model="closed" dt:type="string"/>
</Schema>
<Oblasti xmlns="x-schema:#Schema3">
  <IDOblast>1</IDOblast>
  <Nazov>MURPHOLÓGIA</Nazov>
</Oblasti>
<Oblasti xmlns="x-schema:#Schema3">
  <IDOblast>2</IDOblast>
  <Nazov>APLIKOVANÁ MURPHOLÓGIA</Nazov>
</Oblasti>
```

Vyberať je možné nielen z jednej tabuľky, ale aj z viacerých. Niektoré výsledky dotazov sú v tab. 4.2.

Tab. 4.2. Možnosti klauzuly FOR XML - výber z viacerých tabuliek.

```
SELECT Oblasti.Nazov, Zakony.IDZakon, Zakony.Nazov FROM Zakony, Oblasti
WHERE Oblasti.IDOblast = Zakony.IDOblast AND
(Zakony.IDZakon=7 OR Zakony.IDZakon=627)
FOR XML AUTO

<Oblasti Nazov="MURPHOLÓGIA">
  <Zakony IDZakon="7" Nazov="WHITEHO ZISTENIE"/>
</Oblasti>
<Oblasti Nazov="HLÚPOSŤ">
  <Zakony IDZakon="627" Nazov="MAYNEOV ZÁKON"/>
</Oblasti>
```

```
SELECT Oblasti.Nazov, Zakony.IDZakon, Zakony.Nazov FROM Zakony, Oblasti
WHERE Oblasti.IDOblast = Zakony.IDOblast AND
(Zakony.IDZakon=7 OR Zakony.IDZakon=627)
FOR XML AUTO, ELEMENTS

<Oblasti>
  <Nazov>MURPHOLÓGIA</Nazov>
  <Zakony>
    <IDZakon>7</IDZakon>
    <Nazov>WHITEHO ZISTENIE</Nazov>
  </Zakony>
</Oblasti>
<Oblasti>
  <Nazov>HLÚPOSŤ</Nazov>
  <Zakony>
    <IDZakon>627</IDZakon>
    <Nazov>MAYNEOV ZÁKON</Nazov>
  </Zakony>
</Oblasti>
```

```
SELECT Oblasti.Nazov, Zakony.IDZakon, Zakony.Nazov FROM Zakony, Oblasti
WHERE Oblasti.IDOblast = Zakony.IDOblast AND
(Zakony.IDZakon=7 OR Zakony.IDZakon=627)
FOR XML RAW

Server: Msg 6810, Level 16, State 1, Line 1
Column name 'Nazov' is repeated. The same attribute cannot be generated more than once on the
same XML tag.
```

```
SELECT Oblasti.Nazov as NazovO, Zakony.IDZakon as IDZ, Zakony.Nazov as NazovZ
FROM Zakony, Oblasti WHERE Oblasti.IDOblast = Zakony.IDOblast AND
(Zakony.IDZakon=7 OR Zakony.IDZakon=627)
FOR XML RAW

<row NazovO="MURPHOLÓGIA" IDZ="7" NazovZ="WHITEHO ZISTENIE"/>
<row NazovO="HLÚPOSŤ" IDZ="627" NazovZ="MAYNEOV ZÁKON"/>
```

Vidíme, že tretí z pokusov - `FOR XML RAW` nebol vydarený. Prišlo totiž ku konfliktu mien atribútov. Také konflikty sa dajú riešiť určením názvov výsledných stĺpcov, ako ukazuje posledný príklad v tab. 4.2.

Doterajšie pokusy s výberom údajov z databázy vo formáte XML nám ukazujú, že existujú jasné pravidlá, akými sa vytvárajú prvky, atribúty a ich názvy. Keď tieto pravidlá z nejakého dôvodu neumožňujú získať požadovanú formu výsledkov, je možné použiť mód EXPLICIT. To si však vyžaduje, aby dopyty boli písané podľa určitých pravidiel. Vytvára sa tzv. univerzálna tabuľka s možnosťou definovania názvov, ako aj s možnosťou určenia, či daný údaj má byť zapísaný ako element, alebo atribút. Opis týchto pravidiel je zrejme nad rámec tejto brožúry. Uvedieme však príklad uloženej procedúry, ktorá mód EXPLICIT používa. Táto uložená procedúra rieši výber zákona so zadaným identifikátorom. Výsledok je v podobe, ktorú sme používali v predošlých kapitolách pri práci so súborom Murphyho zákonov. Kód uloženej procedúry je v tab. 4.3.

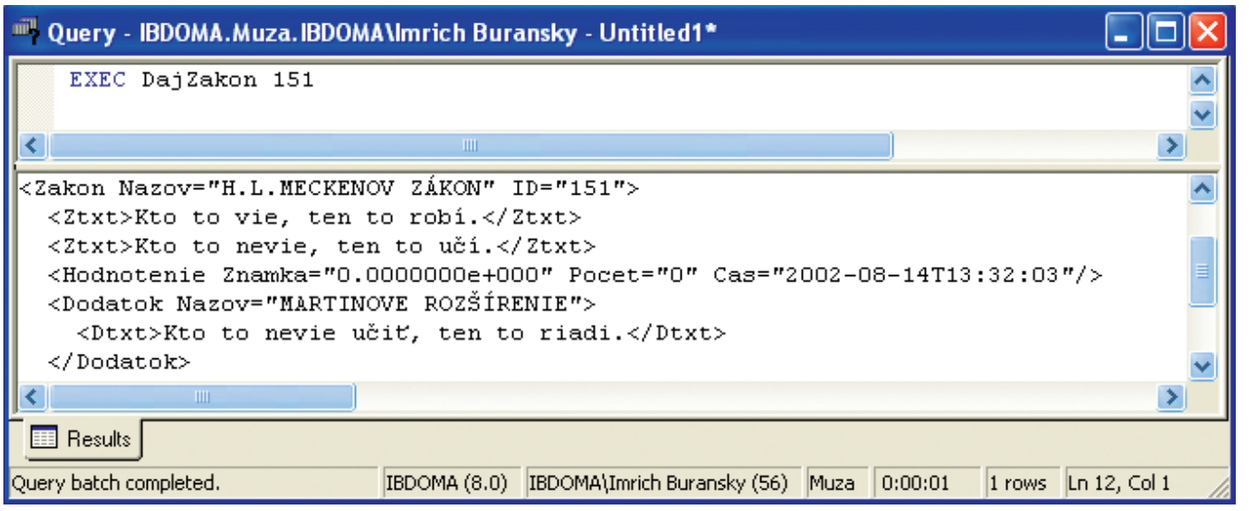
Tab. 4.3. Uložená procedúra DajZakon pre výber Murphyho zákona z databázy Muza.

```
CREATE PROCEDURE DajZakon @ID int
AS

SELECT 1                  as Tag,
          NULL            as Parent,

          Nazov           as [Zakon!1!Nazov],
          IDZakon         as [Zakon!1!ID],
          NULL            as [Ztxt!2!!element],
          NULL            as [Dodatok!3!Nazov],
          NULL            as [Dtxt!4!IDDodatok!hide],
          NULL            as [Dtxt!4!IDTxt!hide],
          NULL            as [Dtxt!4!!element],

          NULL            as [Hodnotenie!5!Znamka],
          NULL            as [Hodnotenie!5!Pocet],
          NULL            as [Hodnotenie!5!Cas]
FROM Zakony
WHERE Zakony.IDZakon=@ID

UNION ALL
SELECT 2 as Tag,
          1 as Parent,
```

```
          NULL,
          IDZakon,
          Txt,
          NULL,
          NULL,
          NULL,
          NULL,

          NULL,
          NULL,
          NULL
FROM  Ztxt
WHERE Ztxt.IDZakon = @ID

UNION ALL
SELECT 3 as Tag,
          1 as Parent,

          NULL,
          IDZakon,
          NULL,
          Nazov,
          IDDodatok,
          NULL,
          NULL,

          NULL,
          NULL,
          NULL
FROM  Dodatky
WHERE Dodatky.IDZakon = @ID

UNION ALL
SELECT 4 as Tag,
          3 as Parent,

          NULL,
          Dodatky.IDZakon,
          NULL,
          NULL,
          Dtxt.IDDodatok,
          Dtxt.IDTxt,
          Dtxt.Txt,

          NULL,
          NULL,
          NULL
FROM Dodatky, Dtxt
WHERE Dodatky.IDZakon = @ID AND Dodatky.IDDodatok = Dtxt.IDDodatok

UNION ALL
SELECT 5 as Tag,
          1 as Parent,

          NULL,
          Hodnotenia.IDZakon,
          NULL,
          NULL,
```

```
          NULL,
          NULL,
          NULL,

          Znamka,
          Pocet,
          Cas
FROM Hodnotenia
WHERE Hodnotenia.IDZakon = @ID
ORDER BY [Zakon!1!ID], [Dtxt!4!IDDodatok!hide], [Dtxt!4!IDTxt!hide]

FOR XML EXPLICIT
GO
```

Výsledok získaný uloženou procedúrou DajZakon je na obr. 4.4.

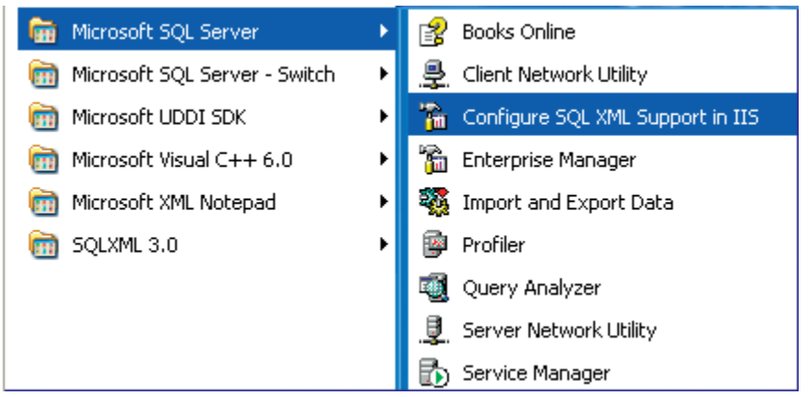


Obr. 4.4. Výsledok získaný uloženou procedúrou DajZakon.

Ak si predstavíme, aké množstvo rôznych údajov je v rôznych databázach a že databázové stroje dokážu dávať výsledky v XML formáte, potom nám je jasná poznámka v predošlej časti, že viac než s tvorbou XML dokumentov sa stretávame s potrebou spracovať údaje v XML formáte.

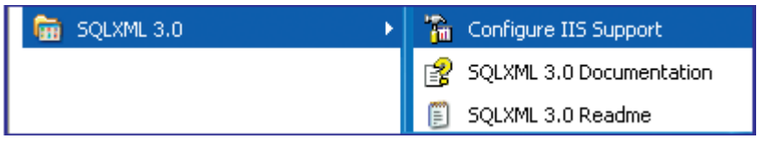
4.2. Konfigurácia SQL XML podpory v IIS

Schopnosť získať údaje v XML formáte priamo z databázového stroja by nebola dôležitá, ak by sme tieto údaje nemohli využiť v sieťovom prostredí. Microsoft SQL Server 2000 to umožňuje s podporou SQLXML a IIS. Začať jeho využívanie treba konfiguráciou. Dostaneme sa k tomu tak, ako ukazuje obr. 4.5.



Obr. 4.5. Kde začať konfiguáciu.

Skôr, než začnete s konfiguráciou, skontrolujte, či máte verziu SQLXML 3.0 SP1. V čase písania tejto brožúry to bola posledná verzia tohto produktu, ktorá v porovnaní s verziou 2.0 poskytuje dôležité rozšírenie pre podporu webových služieb. Ak ju nemáte, dá sa získať na stránke Microsoftu http://microsoft.com/sql/default.asp. A tak radšej začnime tam, ako ukazuje obr. 4.6.



Obr. 4.6. Radšej začnite konfiguráciu SQLXML 3.0 tu.

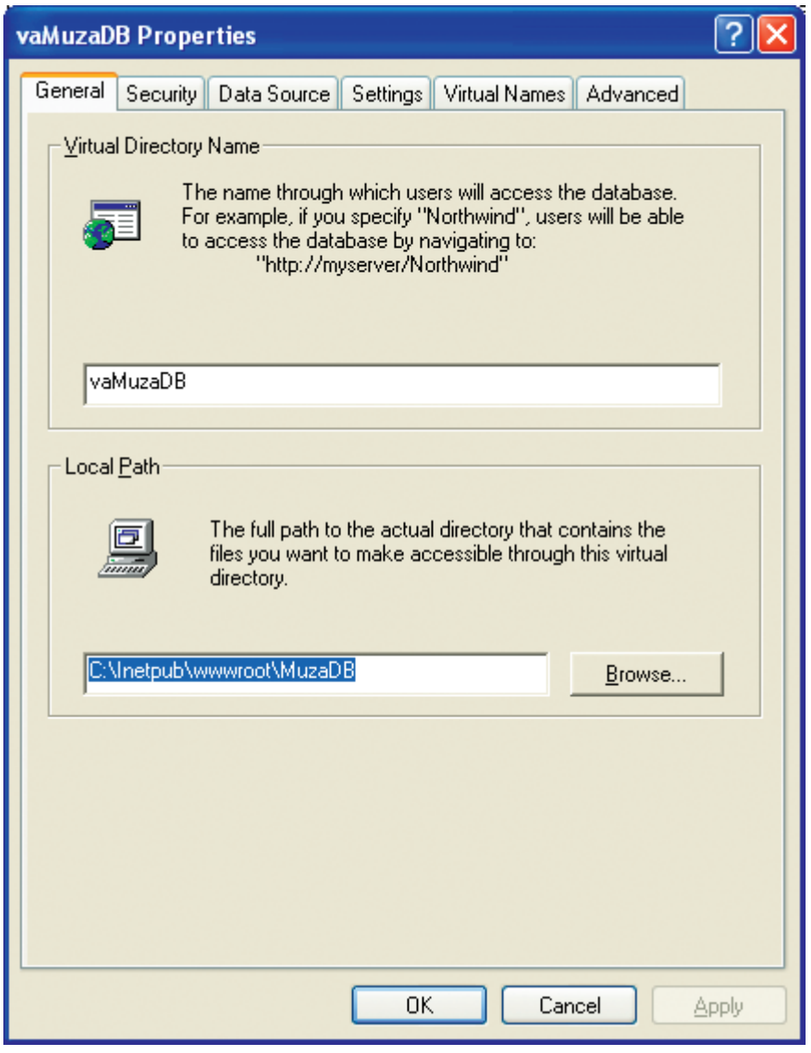
Vytvoríme nový virtuálny adresár, ako ukazuje obr. 4.7.

Obr. 4.7. Vytvorenie virtuálneho adresára.

Nasledujúce obrázky zachytávajú postup konfigurácie virtuálneho adresára vaMuzaDB, ktorý je mapovaný do fyzického adresára počítača C:\Inetpub\wwwroot\MuzaDB. V čase konfigurácie virtuálneho adresára sa samozrejme vyžaduje, aby taký adresár už bol vytvorený. Vytvorme v ňom aj podadresáre :

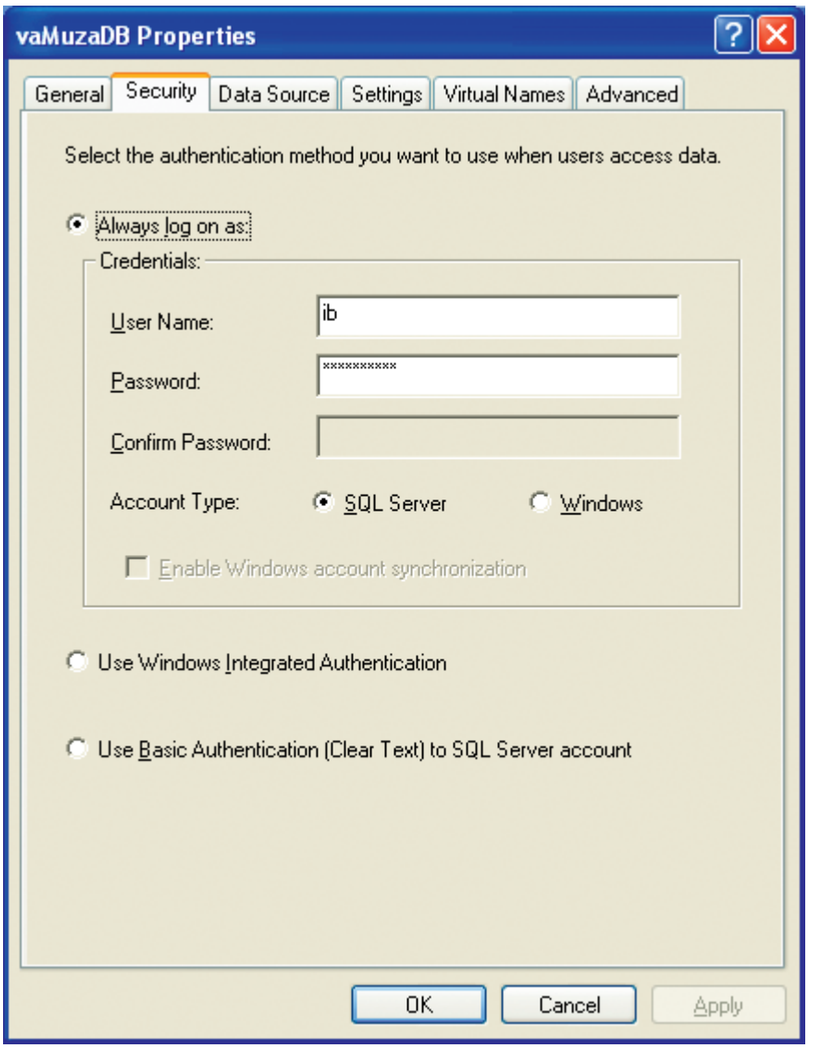
- **template** .. pre šablóny,
- **soap** .. pre opisy webových služieb,
- **dbobject** .. pre databázové objekty,
- **schema** .. pre schémy.

Uvidíte neskôr, že minimálne template a soap budeme potrebovať.



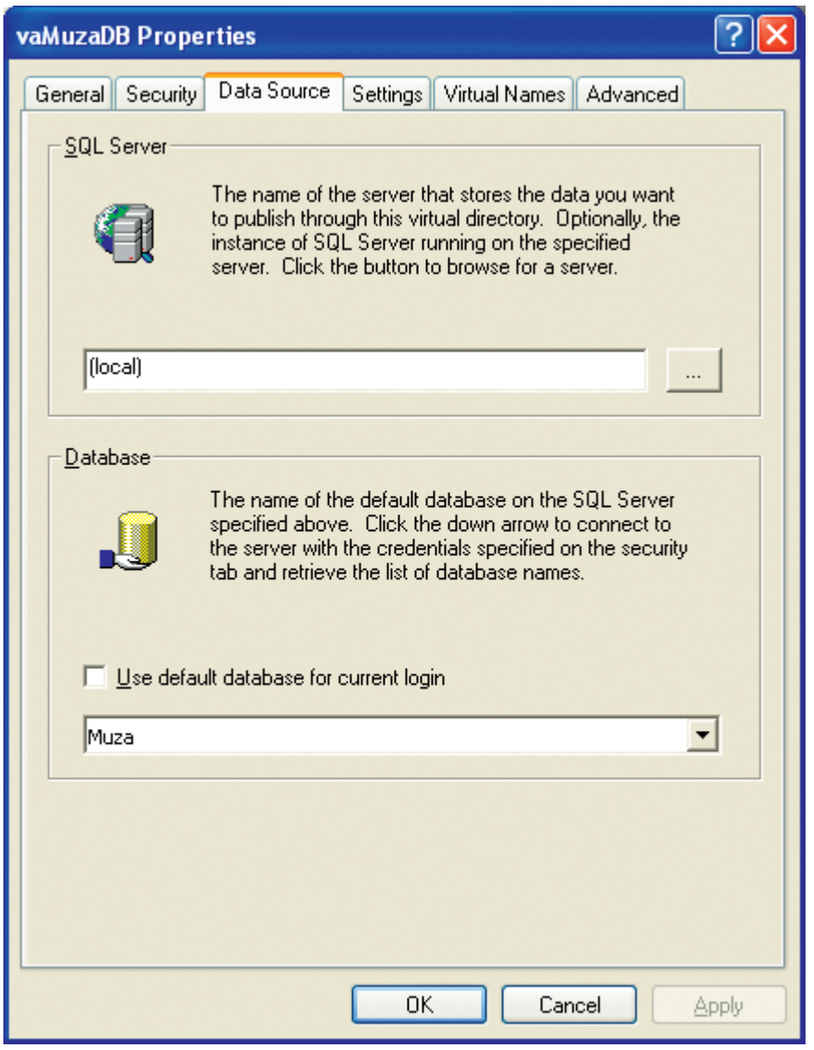
Obr. 4.8. Konfigurácia virtuálneho adresára - záložka General.

Obr. 4.8 ukazuje spôsob zadania názvu virtuálneho adresára vaMuza a jeho mapovanie do fyzického adresára.



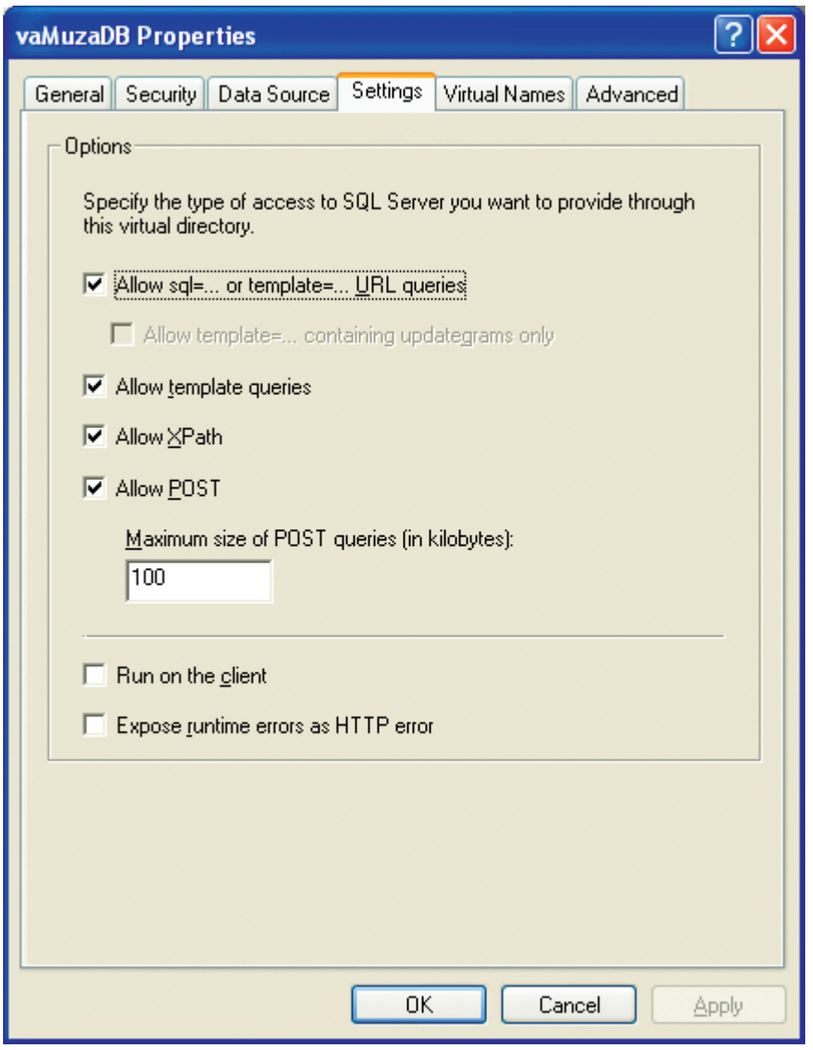
Obr. 4.9. Konfigurácia virtuálneho adresára - záložka Security.

Obr. 4.9. ukazuje voľbu metódy autentifikácie. Pre príklady riešené v tejto brožúre bol v databáze Muza definovaný používateľ ib, s definovanými právami pre prístup k zdrojom databázy. Tomuto používateľovi bol umožnený prístup aj prostreníctvom konfigurovaného virtuálneho adresára.



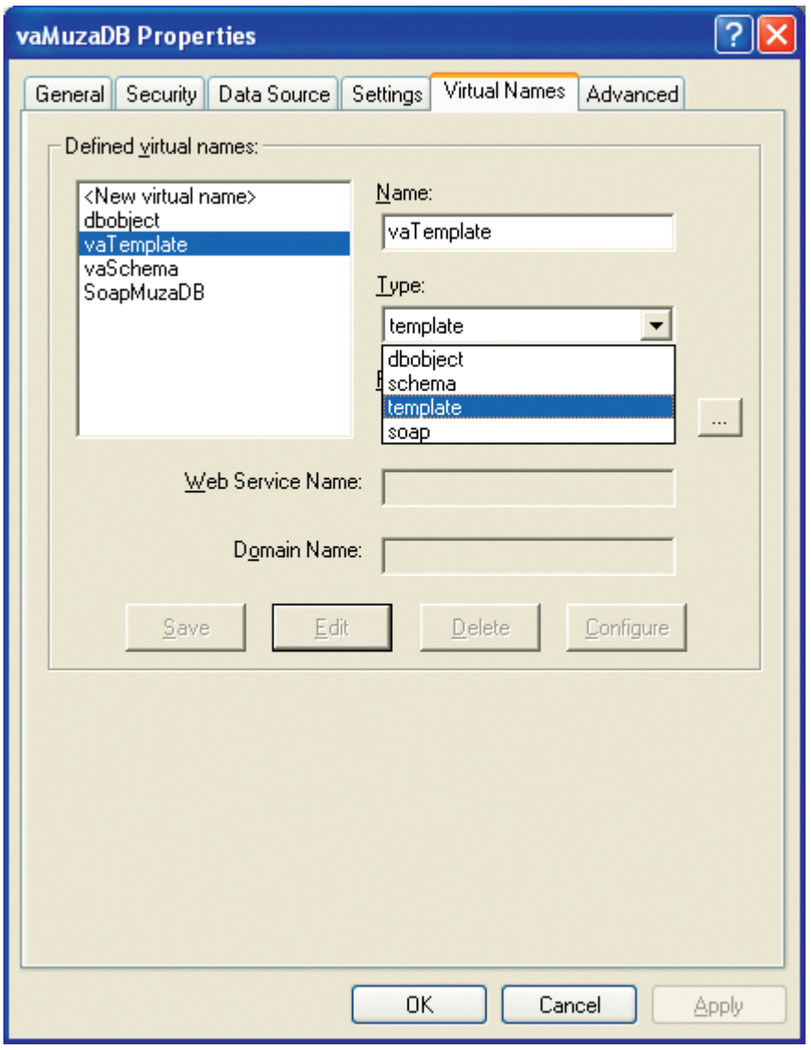
Obr. 4.10. Konfigurácia virtuálneho adresára - záložka Data Source.

Ako naznačuje obr. 4.10, k virtuálnemu adresáru sa priraďuje databáza - v našom prípade je to databáza Muza SQL servera na lokálnom počítači.



Obr. 4.11. Konfigurácia virtuálneho adresára - záložka Setings.

Nastavenia, ktoré ukazuje obr. 4.11 špecifikujú, aký prístup bude umožnený. Pre experimenty je možné použiť viaceré, ale v reálnych podmienkach budete musieť zvážiť bezpečnostné riziká a ponechať iba tie, ktoré budú nevyhnutne potrebné.



Obr. 4.12. Konfigurácia virtuálneho adresára - záložka Virtual Names.

Kým nastavenia na predošlých záložkách majú „dlhodobú“ účinnosť, k záložke s virtuálnymi menami sa budeme častejšie vracať. Tu bude totiž východisko k definovaniu webových služieb. Ale k tomu sa dostaneme v ďalších častiach. Zatiaľ definujme virtuálne mená pre šablóny, schémy dbobjekty a soap. Tu použijeme fyzické adresáre, ktoré sme si pripravili na začiatku.

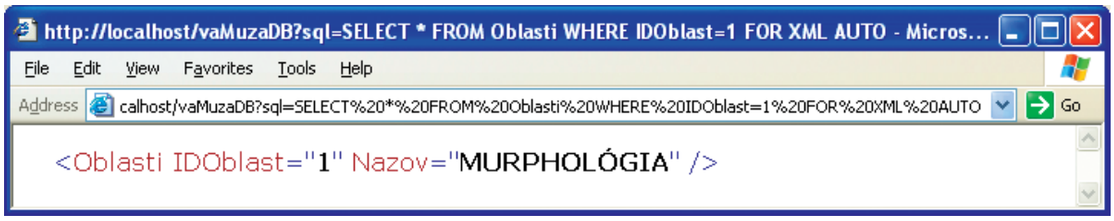
Záložka Advanced poskytuje doplňujúce nastavenia, ale tie v našich experimentoch potrebovať nebudeme.

4.3. Prístup k databáze dopytmi v URL

Na záložke Settings (obr. 4.11) pri konfigurácii SQLXML sme povolili tzv. URL dopyty (URL queries). Poďme vyskúšať, či fungujú. V Internet Exploreri zadajme adresu smerujúcu do vytvoreného virtuálneho adresára. Ako parameter (časť za otáznikom a sql=) zadajme SQL dopyt. Celá adresa bude:

```
http://localhost/vaMuzaDB?sql=SELECT * FROM Oblasti WHERE IDOblast=1 FOR XML AUTO
```

Vidieť, že do adresy sme napísali celý dopyt podobne, ako pri použití SQL Query Analyzer.
Pochopiteľne, využili sme klauzulu FOR XML. Výsledok ukazuje obr. 4.13.



Obr. 4.13. Výsledok prvého dopytu v Internet Exploreri.

Do prehliadača sme získali výsledok priamo z databázy.

Poznamenajme, že trojicou znakov %20 v adrese na obr. 4.13 je nahradzovaná medzera. Robí tak automaticky prehliadač v súlade s pravidlami HTTP protokolu.

Možno vás prekvapilo, že sme zadali požiadavku na výber jedinej oblasti Murphyho zákonov. V dopyte sme definovali obmedzenie WHERE IDOblast=1. Odstráňme túto obmedzujúcu podmienku a zadajme dopyt:

```
http://localhost/vaMuzaDB?sql=SELECT * FROM Oblasti FOR XML AUTO
```

Výsledok bude menej pekný - je na obr. 4.14.

Obr. 4.14. Výsledok druhého dopytu v Internet Exploreri.

Nie že by sme nezískali výsledky z databázy, ale s ponúknutou formou nie je spokojný prehliadač. Upozorňuje nás, že dokument, ktorý má zobraziť, nemá koreňový (vrcholový - top) element. Evidentne však dostávame údaje z databázy.

Musíme vytvoriť opatrenia, ktoré výsledok zabalia tak, aby zodpovedal pravidlám pre správné sformátovanie XML dokumentu.

Východiskom je možnosť definovať koreňový element:

```
http://localhost/vaMuzaDB?sql=SELECT * FROM Oblasti FOR XML AUTO&root=Muza
```

dostaneme takýto výsledok:

```
<?xml version="1.0" encoding="utf-8" ?>
- <Muza>
    <Oblasti IDOblast="1" Nazov="MURPHOLÓGIA" />
   <Oblasti IDOblast="2" Nazov="APLIKOVANÁ MURPHOLÓGIA" />
   <Oblasti IDOblast="3" Nazov="PROJEKTOVANIE" />
   <Oblasti IDOblast="4" Nazov="STROJÁRSTVO" />
   <Oblasti IDOblast="5" Nazov="VEDA, TECHNIKA, VÝSKUM" />
   <Oblasti IDOblast="6" Nazov="RIADENIE" />
   <Oblasti IDOblast="7" Nazov="SCHODZOVANIE" />
   <Oblasti IDOblast="8" Nazov="ÚČTOVNÍCTVO" />
   <Oblasti IDOblast="9" Nazov="ŠPECIALIZÁCIA" />
   <Oblasti IDOblast="10" Nazov="ČLOVEKOVEDA" />
   <Oblasti IDOblast="11" Nazov="METAZÁKONY" />
   <Oblasti IDOblast="12" Nazov="ZÁKONY O VŠEličOM" />
   <Oblasti IDOblast="13" Nazov="HĽADANIE" />
   <Oblasti IDOblast="14" Nazov="OPTIMIZMUS" />
   <Oblasti IDOblast="15" Nazov="KRÍZA" />
   <Oblasti IDOblast="16" Nazov="NOVINY A DOPISY" />
   <Oblasti IDOblast="17" Nazov="KRESLENIE A FOTOGRAFIA" />
   <Oblasti IDOblast="18" Nazov="CESTOVANIE" />
   <Oblasti IDOblast="19" Nazov="BYROKRACIA" />
   <Oblasti IDOblast="20" Nazov="OPRAVY" />
   <Oblasti IDOblast="21" Nazov="ODBORNÍCI" />
   <Oblasti IDOblast="22" Nazov="PLYNUTIE ČASU" />
   <Oblasti IDOblast="23" Nazov="PROBLÉMY" />
   <Oblasti IDOblast="24" Nazov="HLÚPOSŤ" />
</Muza>
```

Problém s koreňovým elementom umožňujú riešiť aj šablóny.

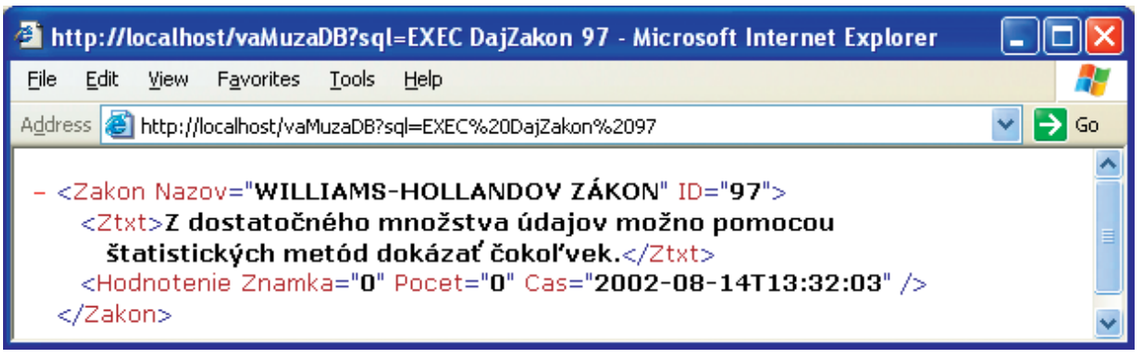
Vytvorme vo fyzickom adresári C:\Inetpub\wwwroot\MuzaDB\template súbor Oblasti1.xml s takýmto obsahom:

```
<root xmlns:sql="urn:schemas-microsoft-com:xml-sql">
<sql:query>
  SELECT * FROM Oblasti FOR XML AUTO
</sql:query>
</root>
```

Je to šablóna, v ktorej je náš dotaz. Zadajme teraz adresu:

```
http://localhost/vaMuzaDB/vaTemplate/Oblasti1.xml
```

Treba si všimnúť, že k šablóne sa dostávame tak, že sme uviedli meno jej virtuálneho adresára - vaTemplate, nie názov fyzického adresára !



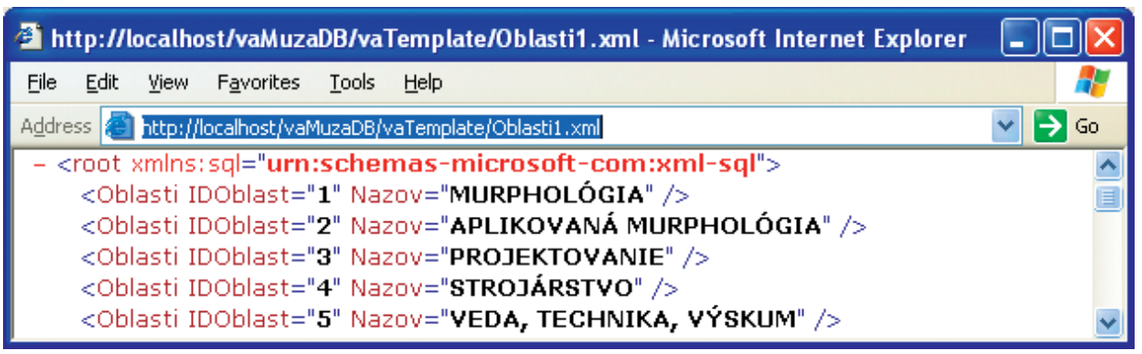
Obr. 4.15. Výsledok volania šablóny Oblasti1.xml.

Vidieť, že výsledok dostávame v XML dokumente s koreňovým elementom root.

Ukážme ešte, že môžeme volať aj uložené procedúry. Zadajme:

```
http://localhost/vaMuzaDB?sql=EXEC DajZakon 97
```

Výsledok ukazuje obr. 4.16



Obr. 4.16. Volanie uloženej procedúry.

Je zrejmé, že uvedené experimenty, ale najmä predvedené možnosti, ako získať údaje z databázy v „rozumnej“ (rozumej XML) forme, dávajú dobré vyhliadky pre naše ďalšie prenikanie do tajomstva XML. A začíname si uvedomovať silu tejto technológie.

4.4. Webová aplikácia s podporou SQLXML

Ukážeme použitie šablón, uložených procedúr a XSL Transformácie pre vyriešenie webovej aplikácie. Budeme v nej chcieť pracovať s Murphyho zákonmi podobným spôsobom, ako v časti 3.4. Rozdiel bude v tom, že nebudeme pracovať s XML súborom, ale údaje budeme čerpať z databázy. Budeme k tomu potrebovať:

- vytvoriť šablónu .. DajZakon.xml,
- vytvoriť XSL transformačný súbor .. MuzaDB.xsl,
- vytvoriť šablónu pre hodnotenie zákona .. Hodnotenie.xml,
- vytvoriť uloženú procedúru pre zápis hodnotenia zákona.. Ohodnotenie.

Šablóna DajZakon.xml je v tab. 4.4. Fyzické umiestnenie súboru je v adresári šablón

```
C:\Inetpub\wwwroot\MuzaDB\template.
```

Tab. 4.4. Šablóna DajZakon.xml.

```
<?xml version='1.0' encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="http://localhost/vaMuzaDB/vaTemplate/MuzaDB.xsl" ?>
<root xmlns:sql="urn:schemas-microsoft-com:xml-sql">
<sql:header>
  <sql:param name="IDZakon">1</sql:param>
</sql:header>
<sql:query>
  EXEC DajZakon @IDZakon
</sql:query>
</root>
```

Vlastný obsah šablóny je v druhom stĺpci. Čísla riadkov v prvom stĺpci slúžia iba pre jednoduchosť odvolávok v opise.

Riadok 1 je prológom XML súboru. Riadok 2 je inštrukcia pre webový prehliadač (Internet Explorer), kde má hľadať súbor s XSL transformáciou. V riadkoch 4 až 6 je hlavička. Je v nej definovaný parameter IDZakon. Jeho hodnota môže byť zadaná v URL pri volaní tejto šablóny. Ak zadaná nebude, vezme sa default hodnota 1 (obsah elementu sql:param). V prvku sql:query - riadky 7 až 9 je SQL dopyt. V našom prípade volanie uloženej procedúry DajZakon s hodnotou parametra IDZakon.

Súbor XSL transformácie MuzaDB.xsl je v tab. 4.5. Aj tento súbor je v adresári šablón `C:\Inetpub\wwwroot\MuzaDB\template`.

Tab. 4.5. Súbor XSL transformácie MuzaDb.xsl.

```
<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match = 'Zakon'>
    <script language="JavaScript"> var ZobrazenyZakon=<xsl:value-of select = '@ID' />;</script>

    <xsl:value-of select = '@Nazov' />

    <xsl:for-each select="Ztxt">  <!-- Pre každý text zákona -->
      <BR /> <xsl:apply-templates />
    </xsl:for-each>                  <!-- Koniec textov zakona -->
    <xsl:for-each select="Dodatok"> <!-- Pre každý dodatok -->
      <I>
      <BR /><BR /><B><xsl:value-of select="@Nazov"/></B>
      <xsl:for-each select="Dtxt">    <!-- Pre kazdy text dodatku -->
        <BR /><xsl:apply-templates />
      </xsl:for-each>                  <!-- Koniec textov dodatku -->
      </I>
    </xsl:for-each>                 <!-- Koniec dodatkov -->
    <BR />
    <xsl:for-each select="Hodnotenie">
      <BR /><B>Hodnotenie:</B>
      známka <B> <xsl:value-of select="@Znamka"/></B>
      počet <B> <xsl:value-of select="@Pocet"/></B>
    </xsl:for-each>

  </xsl:template>

  <xsl:template match = '/'>
    <HTML>
      <HEAD>
        <TITLE>MuzaDB</TITLE>
      </HEAD>

```

```
<script language="JavaScript">
  var ZobrazenyZakon=1;
  function Posun(Kolko)
  {
     var xx;
     xx=(Form1.IDZakon.value)*1+Kolko;
     if (xx>0) { Form1.IDZakon.value=xx; }
     else
     {
       window.alert( "Je zobrazeny prvý zákon! ");
       return false;
     }
     return true;
  }
  function Nahodny()
  {
     Form1.IDZakon.value=Math.round(770*Math.random())+1;
  }
</script>

      <BODY style="font-family: cursive ; background-color: #ffffcc;  text-color: #003300;"
            onload="Form1.IDZakon.value=ZobrazenyZakon;
Form2.IDZakon.value=ZobrazenyZakon;">
          <H2>Murphyho zákony</H2>
          <table width="400" border="0">
          <tr>
            <td><span style="font-size:Smaller">Riadenie výpisu zákonov</span></td>
            <td><span style="font-size:Smaller">Vaše hodnotenie zákona</span></td>
          </tr>
          <tr>
          <td> <form name="Form1" method="post"
action="http://localhost/vaMuzaDB/vaTemplate/DajZakon.xml" >
            <input name="IDZakon" type="text" style="width:49px;" value="1" title="Tu zadajte ID
zákona"  />
            <input type="submit" name="ButDajZakon" value=" !! " title="Vyžiada zákon so zadaným
ID"  />
            <input type="submit" name="ButPredosly" value="&lt;&lt;"   title="Predošlý zákon"
onclick="Posun(-1)" />
            <input type="submit" name="ButDalsi"    value=">>"   title="Ďalší zákon"
onclick="Posun(1)" />
            <input type="submit" name="ButNahodny"  value="??"   title="Náhodný výber zákona"
onclick="Nahodny()" />
          </form> </td>
          <td> <form name="Form2" method="post"
action="http://localhost/vaMuzaDB/vaTemplate/Hodnotenie.xml" >
            <select name="Znamka">
              <option selected="selected" value="1">1 - výborný</option>
              <option value="2">2 - veľmi dobrý</option>
              <option value="3">3 - dobrý</option>
              <option value="4">4 - obstojný</option>
              <option value="5">5 - zlý</option>
            </select>
            <input name="IDZakon" type="hidden"  value="1"  />
            <input type="submit" name="ButNahodny"  value="Odošli"   title="Odoslanie vášho
hodnotenia" />
          </form> </td>
          </tr>
          </table>
```

```
92
93          <xsl:apply-templates select = 'root' />
94       </BODY>
95     </HTML>
96   </xsl:template>
97 </xsl:stylesheet>
98
```

Pri základných poznatkoch z HTML (najmä podstata formulárov), odrobiniek z JavaScriptu, štipky znalostí z XSLT a XPath (boli ponúknuté v časti 3.2) je uvedený transformačný súbor ľahko čitateľný.

Vo výslednom HTML dokumente sú dva formuláre. Form1 obsahuje ovládacie prvky pre „listovanie“ v Murphyho zákonoch (riadky 64 až 76) - zadávacie pole identifikátora zákona IDZakon, tlačidlo pre prechod na zákon so IDZakon - ButDajZakon, ako aj tlačidlá pre zobrazenie predošlého, ďalšieho a náhodného zákona. Formulár Form2 obsahuje prvky potrebné pre ohodnotenie zobrazeného zákona (riadky 77 až 89). Z formulára Form1 je volaná šablóna DajZakon.xml. Z formulára Form2 je volaná šablóna Hodnotenie.xml.

JavaScript je použitý pre odloženie identifikátora zobrazeného Murphyho zákona (riadok 4), pre výpočet posunu vzhľadom na obsah zadávacieho poľa s identifikátorom zákona (riadky 34 až 47) a pre vygenerovanie náhodného čísla (riadky 48 až 51).

Pri XSL transformácii sa vychádza z predpokladu, že transformujeme XML dokument získaný šablónou DajZakon.xml (viď tab. 4.5). V ňom označujeme prvok Zakon (riadok 3). Potom vieme vybrať atribút ID tohto prvku (riadok 4), atribút Nazov - riadok 6. Potom vyberáme obsahy prvkov Ztxt - cyklus začína v riadku 8, končí v riadku 10. Vyberáme tiež dodatky zákona (cyklus v riadkoch 13 až 18). Pre každý dodatok vyberáme jeho texty - prvky Dtxt cyklus v riadkoch 14 až 16. Nakonie pre zákon vyberáme aj jeho hodnotenie - riadky 20 až 24.

Šablóna Hodnotenie.xml je v tab. 4.6. Je veľmi podobná vyššie opísanej šablóne DajZakon.xml. Rozdiel je v tom, že v hlavičke sú dva parametre IDZakon a Znamka (riadky 5 a 6). V prvku sql:query je najskôr volaná uložená procedúra Ohodnotenie (riadok 9), a potom DajZakon (riadok 10). Predpis pre transformáciu získaného XML dokumentu je v MuzaDB.xsl (riadok 2).

Tab. 4.6. Šablóna Hodnotenie.xml.

```
 1 <?xml version='1.0' encoding="utf-8"?>
 2 <?xml-stylesheet type="text/xsl" href="http://localhost/vaMuzaDB/vaTemplate/MuzaDB.xsl" ?>
 3 <root xmlns:sql="urn:schemas-microsoft-com:xml-sql">
 4 <sql:header>
 5   <sql:param name="IDZakon">1</sql:param>
 6   <sql:param name="Znamka">1</sql:param>
 7 </sql:header>
 8 <sql:query>
 9   EXEC Ohodnotenie @IDZakon, @Znamka
10   EXEC DajZakon @IDZakon
11 </sql:query>
12 </root>
```

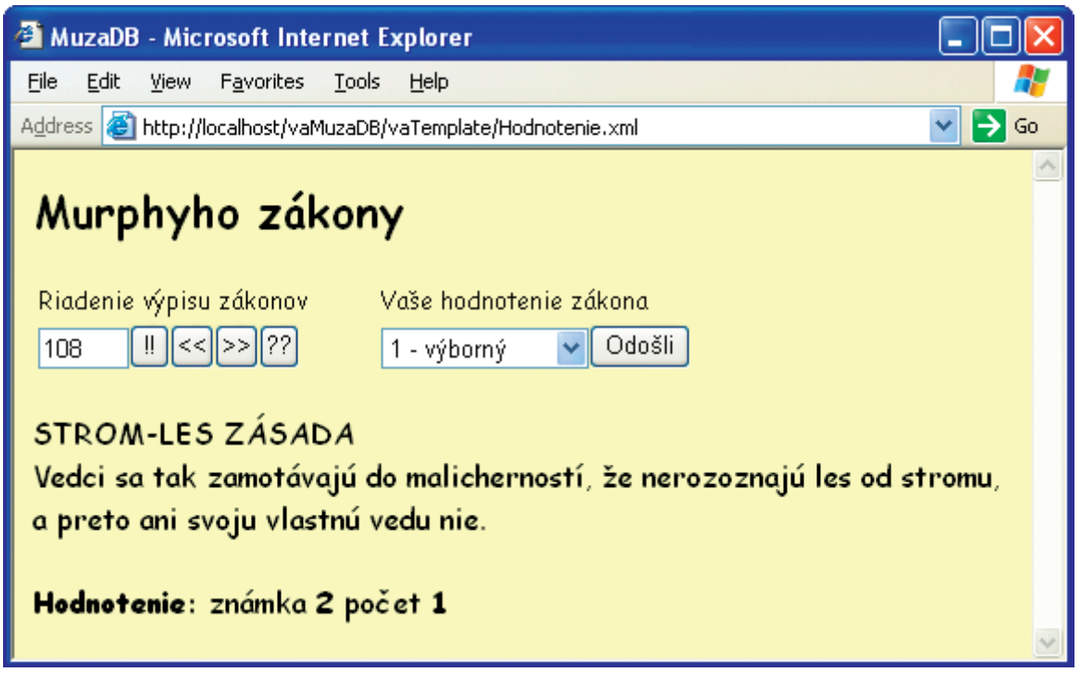
Zostáva nám ukázať **riešenie uloženej procedúry Hodnotenie.** Jej definícia je v tab. 4.7.

Tab. 4.7. Uložená procedúra Hodnotenie databázy Muza.

```
1  CREATE Procedure Ohodnotenie @IDZakon int, @NovaZnamka int
2  AS
3  DECLARE @Znamka real,
4          @Pocet  int,
5          @Cas    datetime;
6  SELECT @Znamka=Znamka, @Pocet=Pocet, @Cas=getdate()
7  FROM Hodnotenia
8  WHERE IDZakon=@IDZakon
9
10 UPDATE Hodnotenia SET
11   Znamka = ((@Znamka*@Pocet + @NovaZnamka)/(@Pocet+1)),
12   Pocet = (@Pocet+1),
13   Cas   = @Cas
14 WHERE IDZakon=@IDZakon
15 GO
```

Procedúra je určená pre pridanie novej známky k označenému zákonu.

Výsledok je webová aplikácia. Veľmi pripomína aplikáciu, ktorú sme opísali v časti 3.4. Ukazuje to obr. 4.17. Spôsob, akým sme sa k nej dopracovali je však úplne iný. Dokonca sme ju získali bez vývojového prostredia Microsoft Visual Studio .NET.



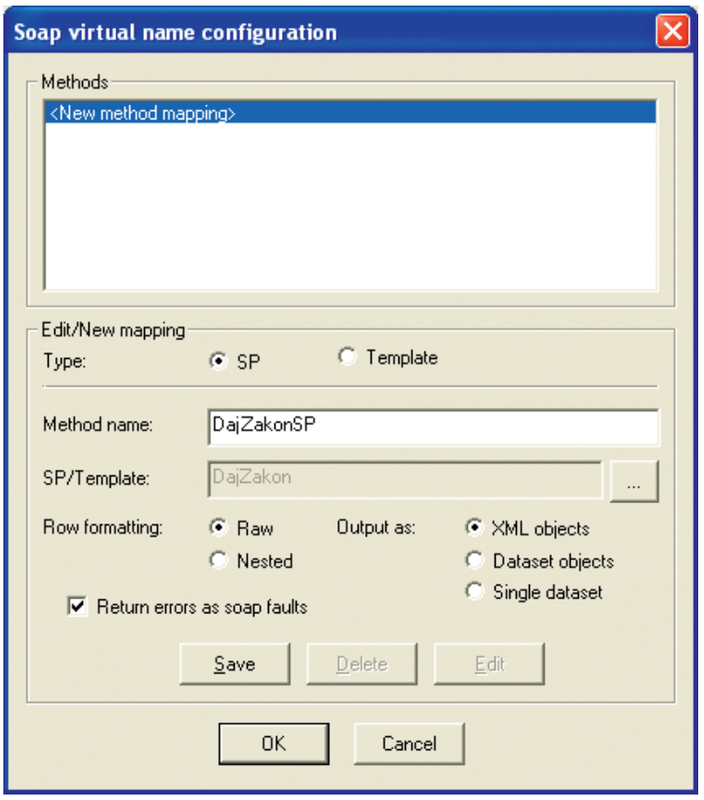
Obr. 4.17. Webová aplikácia vytvorená s podporou SQLXML.

4.5. Konfigurácia webovej služby SoapMuzaDB v SQLXML

Pri opise konfigurácie virtuálneho adresára sme poznamenali, že k záložke Virtual Names sa budeme vracať. Teraz nastala chvíľa, keď budeme v konfigurácii pokračovať. Východisková situácia je:

- V databáze Muza máme uloženú procedúru DajZakon. Pripomeňme, že táto procedúra vydá vo forme XML Murphyho zákon určený zadaným identifikátorom. Kód procedúry je v tab. 4.3.
- V adresári šablón s virtuálnym menom vaTemplates máme súbor DajZakonT.xml. Jeho obsah je modifikáciou šablóny DajZakon.xml z tab. 4.4 Je vynechaný riadok 2. - nebudeme potrebovať XSL transformáciu. V šablóne je volaná uložená procedúra DajZakon.
- Virtuálny adresár typu soap bol pri konfigurácii (obr. 4.12) pomenovaný SoapMuzaDB.

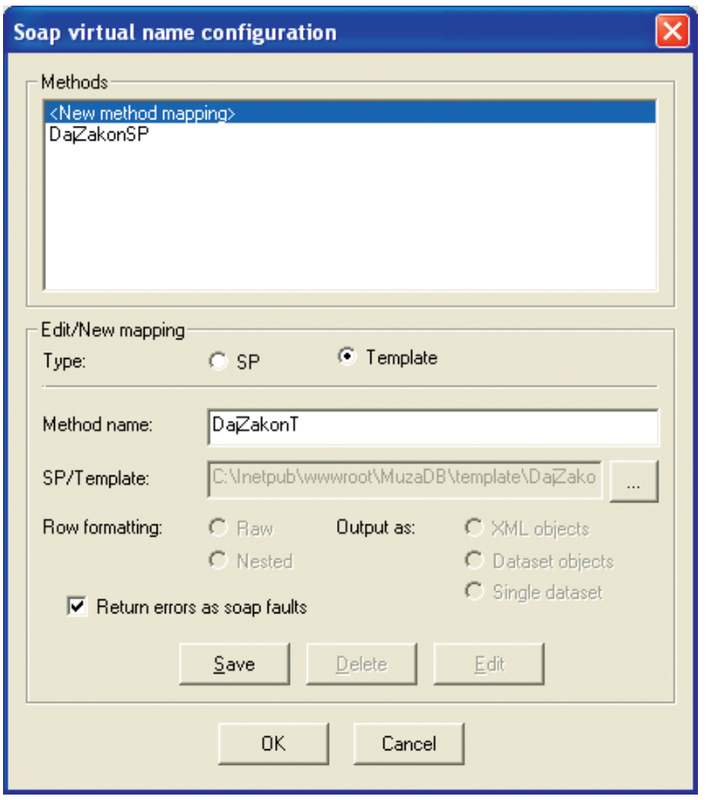
Ku konfigurácii sa dostaneme, tak ako ukazuje obr. 4.6. Zobrazí sa okno podobné obr. 4.7. Bude tam už virtuálny adresár vaMuzaDB. Cez kontextové menu sa dostaneme k vlastnostiam a zobrazíme záložku Virtual Names - ako je na obr. 4.12. Označíme virtuálne meno SoapMuzaDB a zvolíme Configure. Dostaneme možnosť definovať - mapovať metódu, čo zachytáva obr. 4.18



Obr. 4.18. Mapovanie metódy DajZakonSP webovej služby SoapMuzaDB.

Zvolíme nové mapovanie, typ SP (Stored Procedure - uložená procedúra). V riadku SP máme možnost zvoliť tlačidlo ..., čím dostaneme ponuku uložených procedúr. Zvolíme DajZakon. V riadku Method Name môže zadať iné meno, než je meno procedúry. Zvolili sme DajZakonSP, aby sme mali na očiach, že sa jedná o typ mapovania SP. Ostatné voľby umožňujú definovať, v akej podobe budeme získavať výsledok. Zadané údaje uložíme (Save).

Postup si zopakujeme. Teraz budeme robiť mapovanie typu Template. Ukazuje to obr. 4.19.



Obr. 4.19. Mapovanie metódy DajZakonT webovej služby SoapMuzaDB.

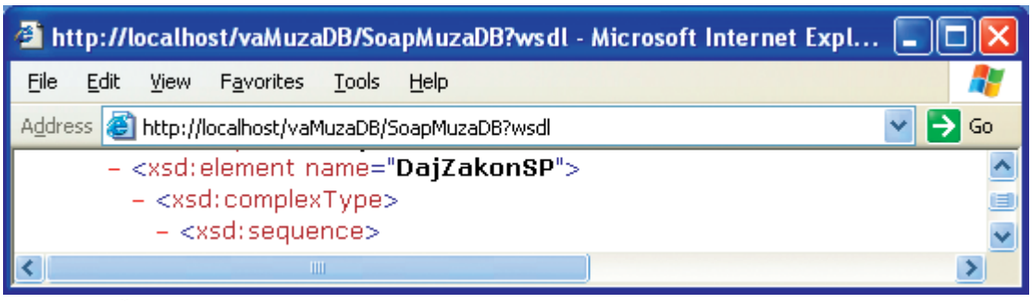
Metóda DajZakonT tvorenej webovej služby je tak mapovaná na šablónu DajZakonT.xml. Uloženie zadaných parametrov mapovania sa odrazí vo vytvorení súboru:

```
C:\Inetpub\wwwroot\MuzaDB\soap\SoapMuzaDB.wsdl
```

Naše doterajšie skúsenosti s webovými službami nám dávajú oprávnenie predpokladať, že sme získali novú webovú službu. Súbory wsdl obsahujú opis webových služieb. Vyššie sme uviedli fyzické umiestnenie súboru. V sieťovom prostredí však musíme pracovať s virtuálnymi adresármi a môžeme rátať s podporou IIS a SQLXML. V internetovom prehliadači zadajme adresu:

```
http://localhost/vaMuzaDB/SoapMuzaDB?wsdl
```

Získame tak dôkaz, že v opise webovej služby SoapMuzaDB sú nami definované metódy. Fragment patriaci k opisu metódy DajZakonSP ukazuje obr. 4.20.



Obr. 4.20. Časť opisu metódy DajZakonSP webovej služby SoapMuzaDB.

Zopakujme si, čo sme pre získanie webovej služby potrebovali.

1. Databázu spravovanú SQL serverom, ktorú chceme využívať v sieťovom prostredí. Máme databázu Muza.
2. Virtuálne adresáre - konfigurácia podpory SQLXML v IIS.
3. Uloženú procedúru a šablónu.
4. Mapovanie metód webovej služby do uložených procedúr, resp. šablón.

Teraz nič nebráni tomu, aby sme vytvorenú webovú službu použili.

4.6. Overenie SoapMuzaDB vo webovej aplikácii KukSoap

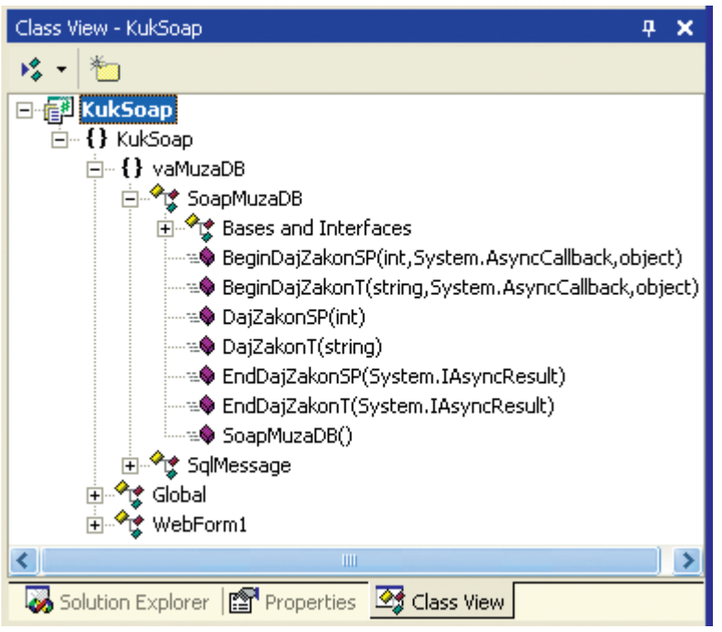
Vytvorme webovú aplikáciu KukSoap pre overenie webovej služby SoapMuzaDB, ktorú sme získali v predošlej časti.

Vo Visual Studiu .NET založme nový projekt typu Visual C#. Použime šablónu ASP .NET Application. Zadajme umiestnenie aplikácie (Location) http://localhost/KukSoap.

Pre možnosť použitia webovej služby SoapMuzaDB treba pridať do projektu webovú referenciu: menu Project - Add Reference. Podobnú úlohu sme riešili v časti 3.6. Zadáme adresu:

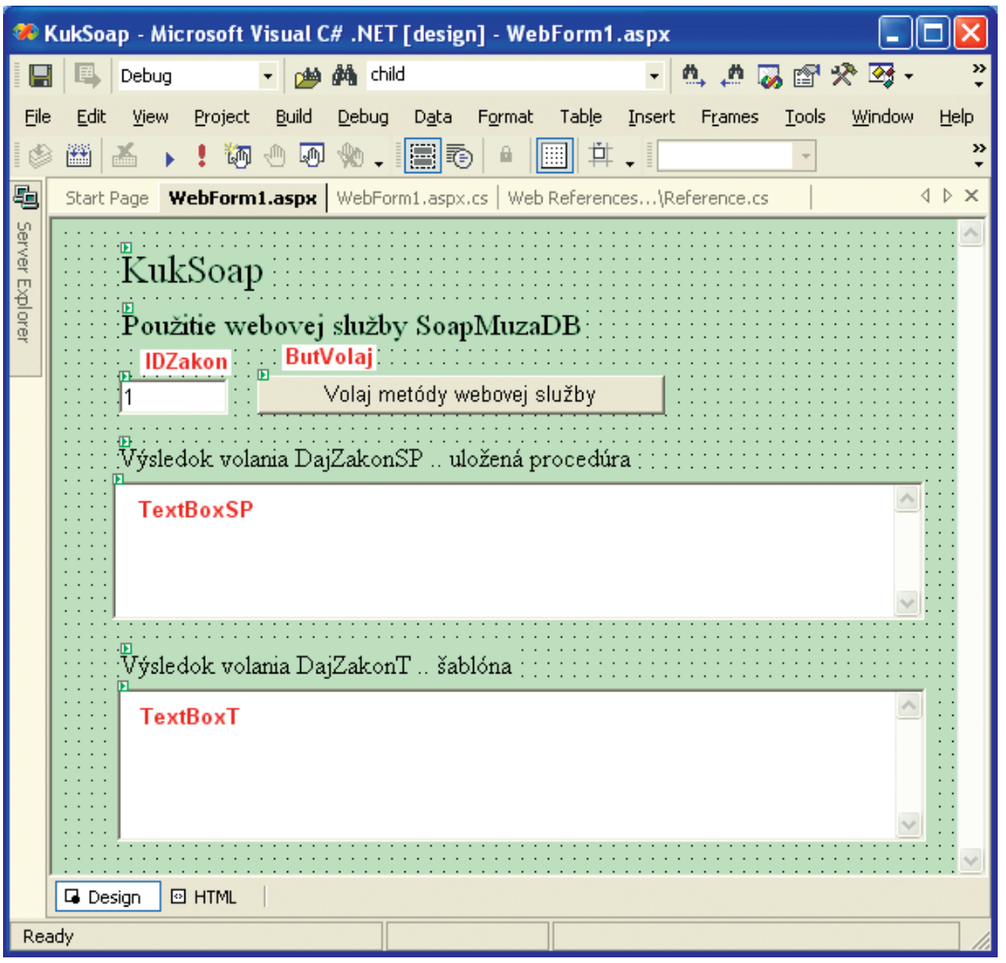
```
http://localhost/vaMuzaDB/SoapMuzaDB?wsdl
```

Po pridaní referencie sa v okne pohľadu na riešenie (Solution Explorer) presvedčíme, že vo webových referenciách je v položke localhost SoapMuzaDB.wsdl. Je to tiež príležitosť, aby sme „localhost“ premenovali na „vaMuzaDB“.
V pohľade na triedy nášho projektu nájdeme stav, ktorý je na obr. 4.21.



Obr. 4.21. Trieda webovej služby SoapMuzaDB.

Vidíme, že trieda má metódu DajZakonSP aj DajZakonT. Všimnite si, že prvá z nich má parameter typu int a druhá typu string. Je to číslo zákona, ktorý budeme žiadať. Budeme chcieť zistiť, čo nám tieto metódy poskytnú.
Navrhnime formulár, ktorý ukazuje obr. 4.22.



Obr. 4.22. Webový formulár aplikácie KukSoap.

Červeným textom sú do formulára doplnené zvolené identifikátory komponentov, s ktorými budeme pracovať:

- IDZakon .. textové pole pre zadanie čísla zákona,
- ButVolaj .. tlačidlo, ktorým budeme aktivovať volanie metód webovej služby,
- TextBoxSP .. textové pole pre výpis výsledkov z volania metódy DajZakonSP,
- TextBoxT .. textové pole pre výpis výsledkov z volania metódy TextBoxT.

Kód, ktorým budeme overovať metódy webovej služby, je v tab. 4.8. Sú tam dve členské metódy triedy WebForm1. Metóda **ButVolaj_Click** je obsluha tlačidla ButVolaj. Metóda **DajInfo** slúži pre prípravu textu informačného výpisu do textových polí. Skôr, než sa pustíme do tvorby kódu, treba preskúmať metódy webovej služby.

Už pri prvom pohľade na triedu, ktorá reprezentuje skúmanú webovú službu, sme zistili, že metódy DajZakonSP a DajZakonT majú parametre rôznych typov (int, string - viď. obr. 4.21). Pri pohľade do zdrojového kódu triedy zistíme, že obe metódy vracajú pole objektov - object []. Postavme si teda cieľ preskúmať tieto polia.

Tab. 4.8. Kód pre overenie webovej služby.

```
1  private void ButVolaj_Click(object sender, System.EventArgs e)
2  { // IB ******************
3    // Vytvorím objekt triedy webovej služby.
4    SoapMuzaDB Muza = new SoapMuzaDB();
5    // Zobrazím informáciu o objektoch získaných z metód.
6    TextBoxSP.Text=DajInfo(Muza.DajZakonSP(Int32.Parse(IDZakon.Text)));
7    TextBoxT. Text=DajInfo(Muza.DajZakonT(IDZakon.Text));
```

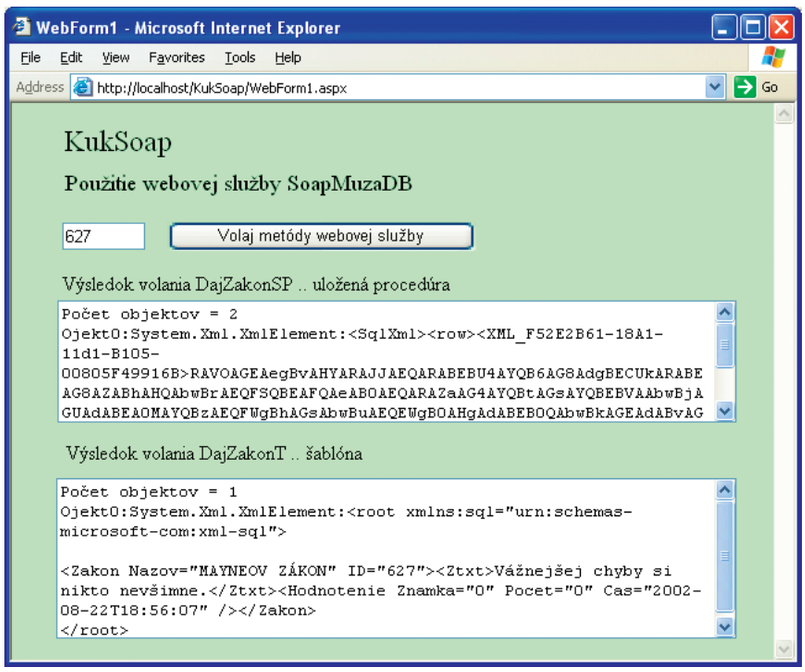
```
 8 }
 9 private string DajInfo (object[] Objekty)
10 { // IB **************
11   string sInfo = "Počet objektov = " + Objekty.Length;
12   for (int i=0; i<Objekty.Length; i++)
13   { string sObjekt=Objekty[i].ToString();
14     sInfo+="\n"+ "Ojekt" + i.ToString() + ":" + sObjekt;
15     // Dostanem XmlElement ?
16     if (sObjekt.CompareTo("System.Xml.XmlElement")==0)
17     { // Áno - mám XmlElement, vypíšem jeho vonkajší text
18       System.Xml.XmlElement e = (System.Xml.XmlElement)Objekty[i];
19       sInfo+=":" +  e.OuterXml;
20     }
21   }
22   return sInfo;
23 }
```

V tab. 4.8. sú komentáre, ktoré vysvetľujú vytvorený kód. Treba poznamenať, že metóda DajInfo je vytvorená tak, aby pripravila text, v ktorom je počet objektov v poli Objekty (riadok 11). Okrem toho, očakávame, že medzi objektmi, ktoré získame z webovej služby, by mohol byť objekt typu `System.Xml.XmlElement`. Ak áno, vypíšeme jeho vonkajší text (riadok 19). Poznamenajme, že vytvorenie objektu typu SoapMuzaDB v riadku 4 je možné za predpokladu, že v priestore mien KukSoap uvedieme using vaMuzaDB.

Použitie webovej aplikácie ukazuje obr. 4.23. Preskúmajme získané výsledky. Pre zjednodušenie opisu budeme používať SP, ak budeme hovoriť o metóde DajZakonSP a T, ak budeme posudzovať metódu DajZakonT webovej služby SoapMuzaDB.

Získané výsledky dokazujú, že metóda SP vracia pole dvoch objektov, zatiaľ čo metóda T iba jeden objekt. Obe metódy vracajú objekt typy `System.Xml.XmlElement`. Ich obsah je však rôzny. Murphyho zákon „je viditeľný“ iba v prípade metódy T. V prípade SP vidíme v získanom texte prvok SqlXml, root a XML_xxx, kde xxx je zrejme text, ktorý má zaručiť jedinečnosť názvu tohto prvku. Murphyho zákon v textovej podobe tam však nie je vidieť.

Uvedený opis získaných výsledkov budeme považovať za východisko k našim ďalším pokusom. Pre vyžiadanie Murphyho zákona z databázy budeme používať metódu DajZakonT webovej služby SoapMuzaDB.



Obr. 4.23. Získaný výsledok z previerky webovej služby SoapMuzaDB.

4.7. Rozšírenie webovej služby SoapMuzaDB

Rozšírme webovú službu SoapMuzaDB tak, aby sme ju mohli použiť vo webovej aplikácii WAMuzaXml. Túto aplikáciu same začali budovať v časti 3.4. Pre prácu s XML súborom Murphyho zákonov sme v nej vytvorili triedu CMuzaXml. Túto triedu sme v časti 3.6 nahradili webovou službou ServiceMuza. Pripomeňme, že webová služba ServiceMuza získava Murphyho zákony zo súboru. Teraz už môžeme povedať, že také riešenie nie je najrozumnejšie. Práca so súborom prináša riziká, ktorým sa vyhneme pri použití služieb databázového stroja. Preto, podľa zásady - do tretice to najlepšie, nahradíme v aplikácii WAMuzaXml webovú službu ServiceMuza službou SoapMuzaDB.

Kým otvoríme projekt WAMuzaXml, urobme „revíziu“ doterajšieho stavu riešenia webovej služby SoapMuzaDB. Zistíme, že nemá ešte všetky metódy, aby mohla byť rovnocennou náhradou za ServiceMuza (viď obr. 3.12). Vie poskytnúť XML dokument Murphyho zákona - metóda DajZakonT, ale nemá metódu DajPocetZakonov, ani PridajZnamku. A tak najskôr vytvoríme tieto metódy. Zopakujeme si pri tom celý postup vytvorenia metódy webovej služby, ktorá je postavená na využití databázového stroja SQL Server 2000 a SQLXML 3.)

Vytvorme metódu DajPocetZakonov. K tomu najskôr definujeme uloženú procedúru databázy Muza. Je v tab. 4.9.

Tab 4.9. Uložená procedúra DajPocetZakonov databázy Muza.

```
1  CREATE Procedure DajPocetZakonov
2  AS
3  SELECT count(*) as Pocet FROM Zakony
4  GO
```

Ak sa chceme vyhnúť nepríjemnostiam, musíme povoliť možnosť vykonávať túto procedúru používateľovi, ktorého sme definovali pri konfigurácii SQLXML podpory na záložke Security (viď obr. 4.9).
Vytvorenú uloženú procedúru použijeme ako základ pre mapovanie do metódy webovej služby. Postupujeme rovnako, ako bolo opísané v časti 4.5 pri definovaní metódy DajZakonSP.

Teraz si asi kladiete otázku, prečo sme za základ pre metódu zobrali uloženú procedúru, keď v predošlej časti sme prišli k záveru, že to „nie je dobré“. Výsledky pre DajZakonSP nedopadli najlepšie (viď obr. 4.23 a komentár k výsledkom). Vysvetlenie nájdeme, ak novovytvorenú metódu DajPocetZakonov overíme podobne, ako sme overovali procedúru DajZakonSP. Dostaneme výsledok, ktorý je v tab. 4.10.

Tab 4.10. Výsledok overovania metódy DajPocetZakonov webovej služby SoapMuzaDB.

```
Počet objektov = 2
Ojekt0:System.Xml.XmlElement:<SqlXml><row><Pocet>770</Pocet></row></SqlXml>
Ojekt1:0
```

Vysvetlenie môže byť v tom, že uložená procedúra DajZakon využíva klauzulu FOR XML. Preto SQL server dáva výsledky vo formáte XML. Vrstva SQLXML takýto výsledok ešte raz zabalí do XML formátu. Výsledok vidíme na obr. 4.23. Údaje z databázy tam zrejme sú, ale ich použitie nie je očividné. Preto sme sa zriekli možnosti použiť metódu DajZakonSP.

V uloženej procedúre DajPocetZakonov nie je klauzula FOR XML. Výsledok do formátu XML balí až SQLXML. Objekt s indexom 0 v poli objektov výsledku je typu System.Xml.XmlElement (viď tab. 4.10). V jeho prvku Pocet nájdeme počet Murphyho zákonov.

Vytvorme metódu PridajZnamku webovej služby SoapMuzaDB. Spomeňme si, že v časti 4.4 sme vytvorili a použili uloženú procedúru databázy Muza s menom Ohodnotenie. Je uvedená v tab. 4.7. Použijeme ju pre mapovanie - zadanie nového virtuálneho mena. To sme už robili viackrát (viď. napr. vytvorenie metódy DajZakonSP v časti 4.5, obr. 4.18).

Presvedčili sme sa, že pre pridanie novej metódy webovej služby s podporou SQL servera 2000 a jeho rozšírenia SQLXML 3. treba:

- definovať uloženú procedúru (nezabudnúť na zadanie oprávnenia pre jej použitie) a/alebo vytvoriť šablónu,
- vytvoriť virtuálne meno, ktorým sa uložená procedúra resp. šablóna mapuje do metódy.

Je užitočné nájsť cestu, ako jednoducho overiť novovytvorenú metódu. Využiteľný je k tomu postup, ktorý sme ukázali v časti 4.6 v podobe webovej aplikácie KukSoap. Pomôže nám overovať čiastočné riešenia. Je to užitočné vtedy, ak nemáme takú možnosť, ako pri tvorbe webovej služby s použitím vývojového prostredia Microsoft Visual Studio .NET, ktorú ukazujú napríklad obr. 3.12 až 3.16.

Opísaný postup si môžeme precvičiť na vytvorení ešte jednej metódy webovej služby SoapMuzaDB. Nazvime ju DajNahodnyZakon. Jej poslaním bude vydať z databázy Muza náhodne vybraný Murphyho zákon vo forme XML.

Definícia uloženej procedúry databázy Muza pre náhodný výber zákona je v tab. 4.11.

Tab. 4.11. Uložená procedúra databázy Muza pre náhodný výber Murphyho zákona.

```
1 CREATE Procedure DajNahodnyZakon
2 AS
3 DECLARE @Pocet int, @IDZakon int
4 SELECT @Pocet = count(*) FROM Zakony
5 SELECT @IDZakon = convert(int,  @Pocet*rand()+1)
6 EXEC DajZakon @IDZakon
7 GO
```

V tejto uloženej procedúre je najskôr zistený počet zákonov (riadok 4) a vygenerované náhodné číslo (riadok 5). Potom je vykonaná procedúra DajZakon (riadok 6).

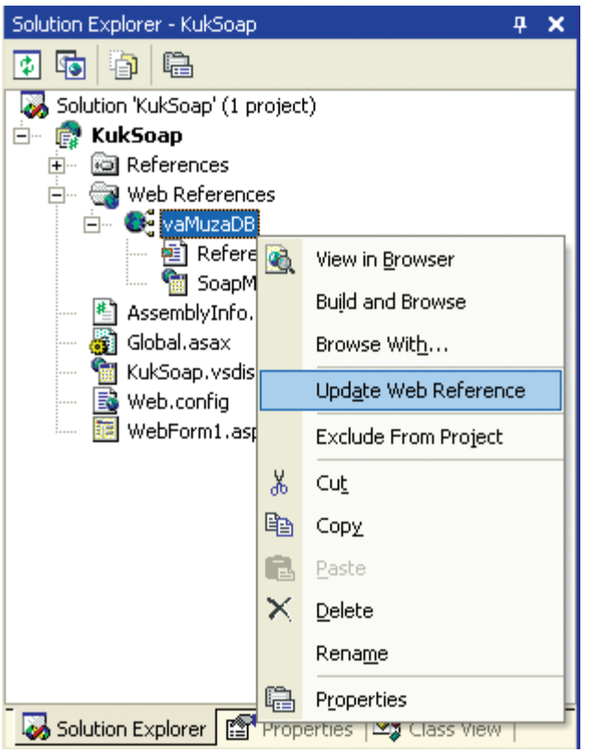
Výsledok z uloženej procedúry DajNahodnyZakon je vo formáte XML, lebo taká je forma výsledku procedúry DajZakon (viď. tab. 4.3). Skúsenosť s takou procedúrou už máme (viď obr. 4.23). Preto nebudeme experimentovať, ale hneď po udelení oprávnenia používať novovytvorenú uloženú procedúru, vytvoríme súbor šablóny - tab. 4.12, v ktorej je volaná uložená procedúra DajNahodnyZakon.

Tab. 4.12. Šablóna pre výber náhodného Murphyho zákona - DajNahodnyZakon.xml.

```
1 <?xml version='1.0' encoding="utf-8"?>
2 <root xmlns:sql="urn:schemas-microsoft-com:xml-sql">
3 <sql:query>
4   EXEC DajNahodnyZakon
5 </sql:query>
6 </root>
```

Konfiguráciu mapovania procedúry do metódy webovej služby považujme už za rutinnú záležitosť. Ak by sme mali problémy, pozrieme sa na postup opísaný v časti 4.5, obr. 4.19, kde je riešený podobný príklad - metóda DajZakonT. Novú metódu nazvime DajNahodnyZakon.

Pre overenie môžeme „zneužiť“ webovú aplikáciu KukSoap. Mierne jej zmeníme výzor a mierne upravíme kód. Predtým však nezabudneme aktualizovať webovú referenciu, ako ukazuje obr. 4.24.



Obr. 4.24. Aktualizácia webovej referencie.

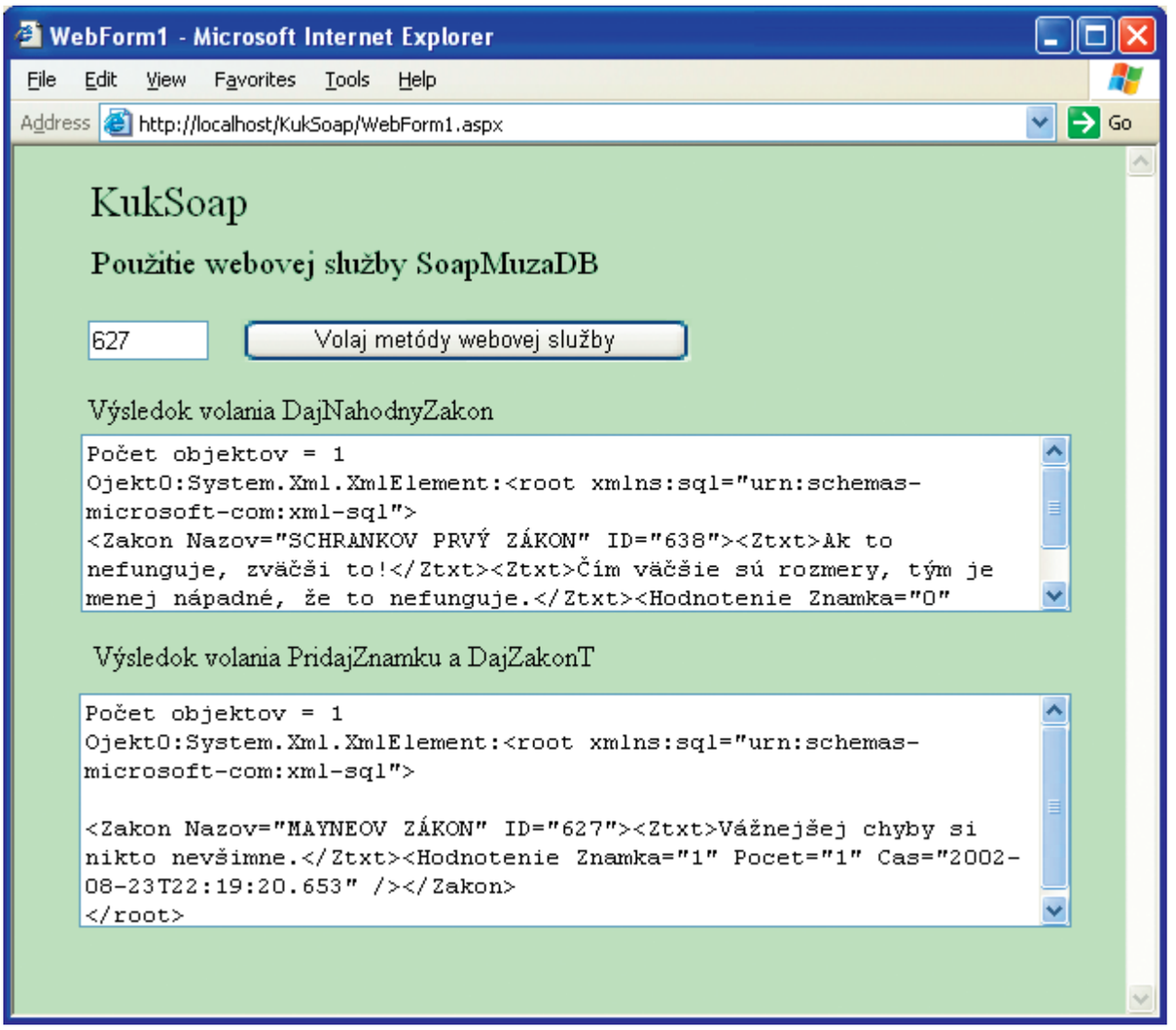
Poznamenajme, že aktualizácia je potrebná, aby sa doplnenie (zmena) metód webovej služby prejavilo v zastupujúcej triede.

Avizovaná drobná úprava kódu aplikácie KukSoap je v tab. 4.13.

Tab. 4.13. Úprava kódu aplikácie KukSoap pre overenie nových metód webovej služby.

```
1 private void ButVolaj_Click(object sender, System.EventArgs e)
2 { // IB ******************
3   // Vytvorím objekt triedy webovej služby.
4   SoapMuzaDB Muza = new SoapMuzaDB();
5   // Zobrazím informáciu o objektoch získaných z metód.
6   TextBoxSP.Text =DajInfo(Muza.DajNahodnyZakon());
7   Muza.PridajZnamku(Int32.Parse(IDZakon.Text),1);
8   TextBoxT. Text =DajInfo(Muza.DajZakonT(IDZakon.Text));
9 }
```

Výsledok vidíme na obr. 4.25.



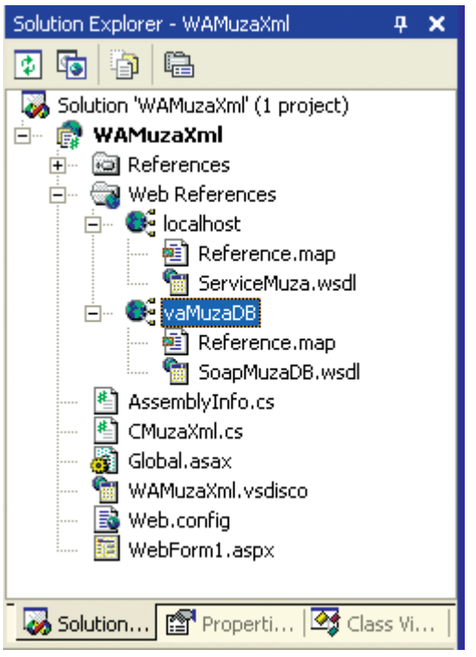
Obr. 4.25. Overenie nových metód webovej služby SoapMuzaDB.

Všimnime si, že náhodne vybraný zákon má ID = 638. Zákonu 627 bola najskôr pridaná známka (jednotka) a až potom bol zobrazený.

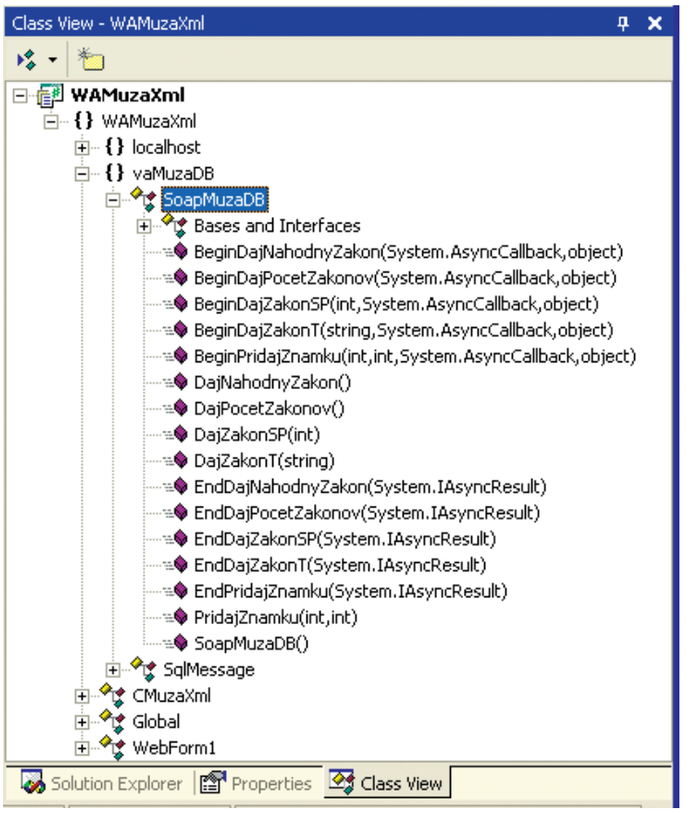
Teraz máme webovú službu SoapMuzaDB vybudovanú tak, že ju môžeme použiť vo webovej aplikácii WAMuzaXML. Navyše máme k dispozíciu metódu DajNahodnyZakon.

4.8. Použitie SoapMuzaDB vo webovej aplikácii WAMuzaXml

Otvorme projekt WAMuzaXml. Pridajme do neho referenciu na webovú službu SoapMuzaDB. Postupujeme tak, ako v 4.6 pri pridávaní referencie na túto službu v aplikácii KukSoap. Aj tu zmeníme meno referencie na vaMuzaDB. V projekte budeme mať referencie dvoch webových služieb - obr. 4.26.



Obr. 4.26. Pohľad na referencie dvoch webových služieb.



Obr. 4.27. Trieda zastupujúca webovú službu SoapMuzaDB.

Preskúmame triedu, ktorá v našom projekte zastupuje webovú službu SoapMuzaDB - obr. 4.27. Vidíme, že má aj metódy, ktoré sme do služby pridali v predošlej časti.

Použitie triedy SoapMuzaDB vo WebForm1.asmx.cs začneme definíciou použitia priestoru mien:

```
using vaMuzaDB;
```

Tab. 4.14. Použitie webovej služby SoapMuzaDB.

```
 1 public void ZobrazZakon(int iz)
 2 { // IB ***************
 3   SoapMuzaDB MuzaXml = new SoapMuzaDB();
 4   object[] PoleObjektov=MuzaXml.DajZakonT(iz.ToString());
 5   XmlElement x= (XmlElement)PoleObjektov[0];
 6   // Získaný element budem transformovať.
 7   XslTransform xslt = new XslTransform();
 8   // Zavediem transformačný súbor.
 9   xslt.Load(SuborXSL);
10   // Vytvorím objekt, do ktorého bude zapísaný výsledok transformácie.
11   System.IO.StringWriter swriter = new System.IO.StringWriter();
12   xslt.Transform(x,null,swriter);
13   // Vypíšem výsledok transformácie.
14   LabelZakon.Text=swriter.ToString();
15   // Vypíšem aj číslo zákona.
16   IDZakona.Text=iz.ToString();
17 }
18 private void ButNahodnyVyber_Click(object sender, System.EventArgs e)
19 { // IB *************************
20   SoapMuzaDB MuzaXml = new SoapMuzaDB();
21   object[] PoleObjektov=MuzaXml.DajPocetZakonov();
22   XmlElement x = (XmlElement)PoleObjektov[0]; // <SqlXml>
23   XmlNode n1 =  x.FirstChild;                  //   <root>
24   XmlNode n2 = n1.FirstChild;                  //     <Pocet
25   int nZakonov = Int32.Parse(n2.InnerText);
26   ZobrazZakon(GeneratorCisel.Next(1,nZakonov));
27 }
28 private void ButZnamka_Click(object sender, System.EventArgs e)
29 { // IB *******************
30   int iz = Int32.Parse(IDZakona.Text);
31   int Znamka = DropDownZnamka.SelectedIndex+1;
32   { SoapMuzaDB MuzaXml = new SoapMuzaDB();
33     MuzaXml.PridajZnamku(iz, Znamka);
34   }
35   ZobrazZakon(iz);
36 }
```

Porovnajme nové riešenie s pôvodným, ktoré je v tretej kapitole. V tab. 3.6 a 3.8 je riešenie s použitím triedy CMuzaXml. V tab. 3.10 je ukázaná náhrada CMuzaXml webovou službou ServiceMuza. V tab. 4.14. je riešenie s webovou službou SoapMuzaDB. Zmeny sa týkajú metódy ZobrazZakon, ButNahodnyVyber_Click a ButZnamka_Click.

V metóde ZobrazZakon je zásadný rozdiel v tom, že kým v pôvodnom riešení získavame údaje zákona v textovom reťazci, v novom sú tieto údaje v poli objektov. V nultom prvku tohto poľa je objekt typu XmlElement, ktorý získame v riadku 5. Použijeme ho aj v XSL transformácii - riadok 12.

V metóde ButNahodnyVyber je potrebné získať počet zákonov. Využíva sa k tomu metóda DajPocetZakonov webovej služby SoapMuzaDB (riadok 21). Z overenia tejto metódy vieme, že výsledok dostávame v objekte typu XmlElemet. V ňom je hierarchia prvkov `<SqlXml><root><Pocet>xxx</Pocet></root></SqlXml>` (viď tab. 4.10). Našou úlohou je získať vnútorný text prvku Pocet, kde na mieste xxx je počet zákonov. Dosiahneme to dvojnásobným volaním FirstChild (property objektov tried XmlElement a XmlNode). Je to v riadkoch 23 a 24. Získame tak objekt n2 typu XmlNode, ktorého InnerText je hľadaný počet zákonov. Text transformuje na číslo - riadok 25.

V metóde ButZnamka_Click v porovnaní s pôvodným riešením nie sú žiadne zvláštnosti.

Výsledok opísaných úprav webovej aplikácie WAMuzaXml nemá žiadne dopady na jej vonkajší prejav - používateľské rozhranie. Murphyho zákon bude zobrazený v podobe, ktorá je na obr. 3.7. Odstránili sme však „rizikové“ použitie XML súboru. Namiesto toho údaje získavame z databázy.

P.S.

V používateľskom rozhraní je predsa len drobná odlišnosť. Ak zadáme neexistujúce IDZakona, nedočkáme sa oznamu o chybe, ako ukazuje obr. 3.8. Nedostatok môžeme odstrániť, ak v tab. 4.14 za riadok 14 vložíme:

```
if (LabelZakon.Text.IndexOf("Hodnotenie")<0)
   LabelZakon.Text="XXX CHYBA XXX <BR />Nepodarilo za získať zákon " +
   iz.ToString() + " ;-(";
```

5. Dáta posielané webovou službou

Motto: FINAGLEHO PRVÝ ZÁKON
Ak dôkazom zistíš, že pokus bol správny, tak si v ňom musel urobiť chybu.

V doterajších experimentoch s webovými službami sme získavali dáta v podobe jednoduchých typov. Počet Murphyho zákonov sme dostávali ako celé číslo. Vlastný zákon sme prenášali v textovom reťazci. Využili sme však pri tom zápis zákona pomocou XML. S jednoduchým typom, ako je int, je to naozaj jednoduché. Horšie je to so „zložitým“ typom skrytým v textovom reťazci. Aj napriek tomu, že Microsoft .NET Framework ponúka široké možnosti pre prácu s údajmi vo formáte XML, dolovanie údajov z textového reťazca nie je jednoduché. Programátor musí poznať možnú štruktúru spracovávaného XML dokumentu. Potrebuje to pri priamej manipulácii s XML dokumentom (viď napr. tab. 3.4), aj pri tvorbe transformačného predpisu pre zobrazenie údajov (viď napr. tab. 3.7). Úlohu mu neuľahčí ani to, ak údaje od webovej služby získa v objekte typu System.Xml.XmlElement (viď napr. tab. 4.14).

Ukázali sme už, ako jednoducho je možné zobraziť XML dokument, ak ho načítame do objektu typu System.Data.DataSet a ten odovzdáme datagridu (viď obr. 2.15). Pokúsime sa to urobiť aj s XML textom Murphyho zákona, ktorý získame z webovej služby. Ukážeme tiež, ako vyslať z webovej služby priamo dataset.

Možnosti pre zefektívnenie práce s údajmi ponúka XML serializácia. Ona je v pozadí toho, že volaním metódy webovej služby je možné získať objekt. V jeho dátových členoch sú údaje poskytované webovou službou. Znamená to, že vo webovej službe namiesto textového reťazca vytvárame objekt, ktorý je návratovým typom jeho metódy.

5.1. Založenie projektu Muza4

Doterajšie prenikanie do hĺbin XML sme robili na konkrétnych príkladoch. Aj nasledujúce pokusy budeme robiť v konkrétnej aplikácii. V Microsoft Visual Studio .NET vytvorme nový projekt typu C# so šablónou Windows Application. Nazvime ho Muza4. Budeme v nej používať webovú službu WSMuzaXml - jej vytvorenie bolo opísané v časti 3.5, aj webovú službu SoapMuzaDB vytvorenú v štvrtej kapitole s podporou SQLXML. Nám už známym spôsobom vyvoláme menu Project - Add Web Reference. Zadáme adresy webových služieb:

```
http://localhost/WSMuzaXml/ServiceMuza.asmx?WSDL
```

```
http://localhost/vaMuzaDB/SoapMuzaDB?WSDL
```

Pre lepšiu orientáciu v projekte premenujme ponúknuté pomenovanie referencie localhost resp. localhost1 na WSMuzaXml resp. vaMuzaDB. Urobíme to v okne prieskumníka riešenia - Solution Explorer. Poznamenajme, že tieto webové služby budeme ešte v našich pokusoch meniť. Po ich zmene vyvoláme Update Web Reference.

V návrhovom zobrazení umiestnime do rámca aplikácie datagrid - pomenujme ho dataGridMuza. Umiestnime tam aj tlačidlo - pomenujme ho ButDajXml. Poznamenajme, že používateľské rozhranie aplikácie Muza4 budeme ešte rozširovať. Opísaný stav je dostatočný na to, aby sme mohli začať experimenty s objektmi typu DataSet.

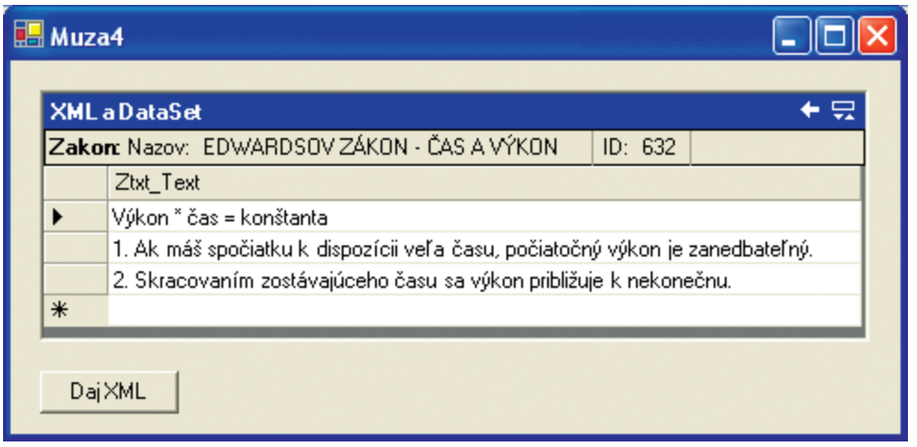
5.2. Vytvorenie objektu typu DataSet na strane klienta

Vyvolajme metódu DajZakon webovej služby WSMuzaXml. Získaný textový reťazec odovzdajme objektu typu DataSet a zobrazme ho v datagride. Opísanú úlohu budeme riešiť v obsluhe stlačenia tlačidla ButBajXml. Riešenie je v tab. 5.1.

```
 1 private void ButDajXml_Click(object sender, System.EventArgs e)
 2 { // IB ********************
 3   ServiceMuza Muza = new ServiceMuza();
 4   string sZakon=  Muza.DajZakon(632);
 5   DataSet dsMuza = new DataSet();
 6   dsMuza.ReadXml(new XmlTextReader(new System.IO.StringReader(sZakon)));
 7   dataGridMuza.DataSource = dsMuza;
 8   dataGridMuza.DataMember = "Zakon";
 9   dataGridMuza.CaptionText = "XML a DataSet";
10 }
```

Tab. 5.1. Využite XML textu v objekte typy DataSet.

V riadku 3 je vytvorená inštancia webovej služby. Vyžiadame od nej zákon s identifikátorm 632. Získaný text máme v premennej sZakon. Vytvoríme objekt typu DataSet (riadok 5 - premenná dsMuza) a prinútime ho načítať získaný text zákona (riadok 6). Ako dôkaz, že sa nám to podarilo, necháme dsMuza zobraziť v data gride. Výsledok ukazuje obr. 5.1.



Obr. 5.1. Zákon odovzdaný do datagridu objektom typu DataSet.

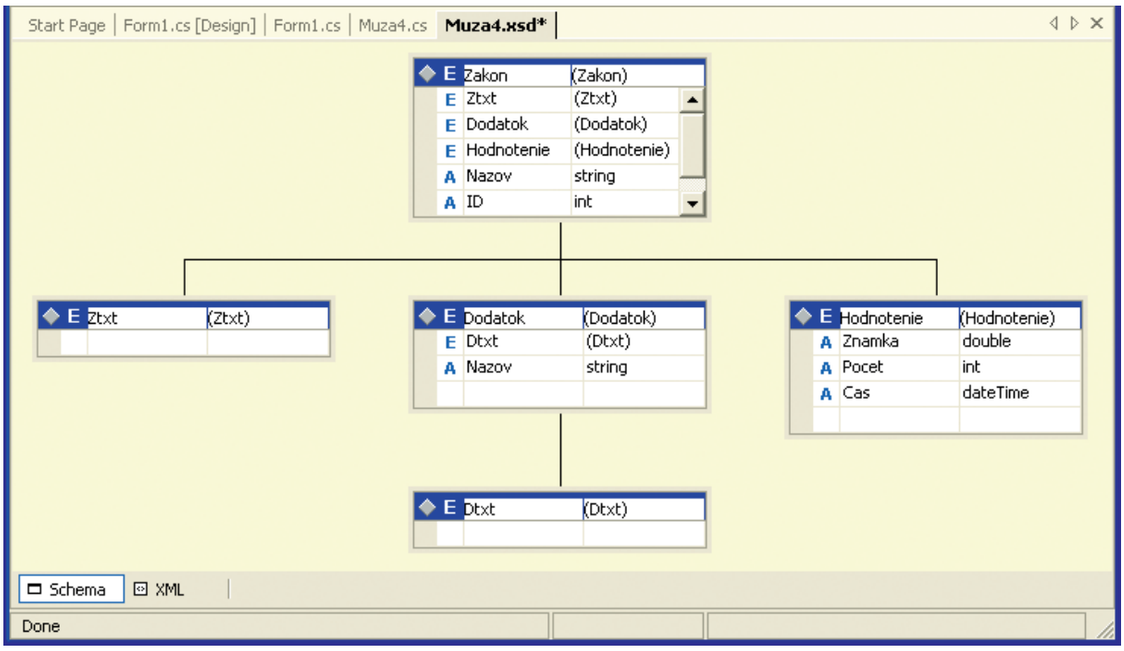
Je zrejmé, že ak máme objekt typu DataSet, môžeme využiť všetky jeho vlastnosti a metódy. Môžeme tak manipulovať s údajmi, ktoré obsahuje. Taká manipulácia však bude oveľa efektívnejšia, ak môžeme uvažovať nie so všeobecným XML textom, ale s textom, v ktorom očakávame Murphyho zákon. Aby sme to dosiahli, budeme postupovať takto:

- Vytvoríme „prototyp“ XML textu, ktorý budeme spracovávať.
- Vytvoríme XML schému, ktorá bude taký text definovať.
- Zo schémy vygenerujeme data set.

Samozrejme, prvý bod môžeme vynechať, ak naše schopnosti vytvárať XML schému sú postačujúce na to, aby sme sa pustili priamo do budovania schémy.

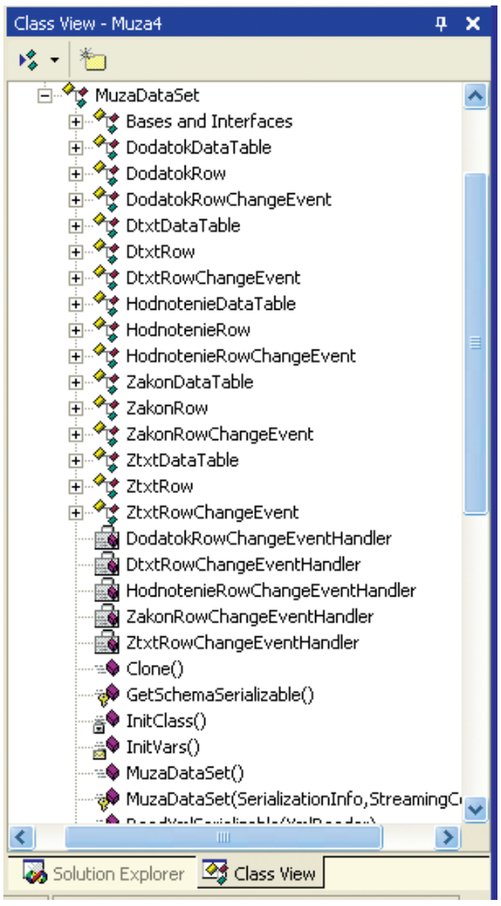
Prototyp XML textu vytvoríme tak, že do súboru zapíšeme jeden zákon, napríklad ten s identifikátorom 632, ktorý sme použili vyššie. Zdrojom nám môže byť súbor Murphyho zákonov, ktorý sme vytvorili v časti 2.6. Zvolený zákon zapíšme do súboru Muza4.xml a umiestnime ho v adresári projektu našej aplikácie Muza4. Zaraďme tento súbor do projektu. Použijeme k tomu menu Project - Add Existing Item. Ak potom zobrazíme tento súbor, získame možnosť vytvoriť XML schému. Východiskom bude podobná situácia, akú ukazuje obr. 2.6.

Získanú XML schému budeme musieť upraviť. Tak napríklad, identifikátor zákona - atribút ID v nej bude typu string, ale my by sme mohli žiadať pracovať s typom int. Podobne upravíme aj typy atribútov hodnotenia zákona - Znamka double, Pocet int, Cas dateTime. Je potrebné nastaviť aj atribút targetNamespace schémy. V našom príklade je http://buransky.sk/Muza4.xsd. Nastavme tiež vlastnosť id schémy na MuzaDataSet. Výsledná schéma je na obr. 5.2.



Obr. 5.2. XML schéma jedného Murphyho zákona.

Z kontextového menu pri zobrazení XML schémy vyvoláme Generate Dataset. Vytvoríme tak novú triedu MuzaDataSet odvodenú od DataSet. Čiastkový pohľad na ňu zachytáva obr. 5.3.



Obr. 5.3. Pohľad na triedu MuzaDataSet.

Postavme si teraz cieľ odovzdať textový reťazec Murphyho zákona získaný z webovej služby nie objektu typu DataSet, ale objektu typu MuzaDataSet. Núka sa riešenie, ktoré je v tab. 5.1. V ňom by bolo treba zmeniť riadok 5. Ukazuje sa však, že až také jednoduché to nebude. Žiadaného výsledku - podobný, ako je na obr. 5.1, sa nedočkáme. Príčina je v tom, že v texte, ktorý dostávame z webovej služby, nie je definovaný priestor mien - http://buransky.sk/Muza4.xsd. V kóde v tab. 5.2 je to doplnené.

Tab. 5.2. Využite XML textu v objekte typy MuzaDataSet.

```
 1 private void ButDajXml_Click(object sender, System.EventArgs e)
 2 { // IB ********************
 3   ServiceMuza Muza = new ServiceMuza();
 4   string sZakon=  Muza.DajZakon(632);
 5   MuzaDataSet dsMuza = new MuzaDataSet();
 6   string sDoc = "";
 7   int nz=sZakon.Length;
 8   if (nz>7)
 9   { sDoc += sZakon.Substring(0, 7);
10     sDoc += "xmlns='http://buransky.sk/Muza4.xsd'";
11     sDoc += sZakon.Substring(6, nz-6);
12   }
13   dsMuza.ReadXml(new XmlTextReader(new System.IO.StringReader(sDoc)));
14   dataGridMuza.DataSource = dsMuza;
15   dataGridMuza.DataMember = "Zakon";
16   dataGridMuza.CaptionText = "XML a MuzaDataSet";
17 }
```

Výsledok zobrazený v datagride je podobný tomu, ktorý vidieť na obr. 5.1. Pre manipuláciu s údajmi v objekte typu MuzaDataSet však máme k dispozícii členské premenné a metódy „ušité na mieru“ Murphyho zákonom.

5.3. Vytvorenie objektu typu DataSet na strane servera

Vo webovej službe vytváranej na strane servera s využitím SQLXML 3.0 vytvorenie objektu typu DataSet je otázkou zadania príslušných konfiguračných údajov. Postupujeme pri tom tak, ako bolo opísané v štvrtej kapitole. Treba:

- vytvoriť uloženú procedúru,
- urobiť mapovanie uloženej procedúry do metódy webovej služby.

Vytvorme uloženú procedúru databázy Muza, ktorá vydá zoznam všetkých evidovaných Murphyho zákonov. Je kód je v tab. 5.3.

Tab. 5.3. Uložená procedúra databázy Muza pre vydanie zoznamu Murphyho zákonov.

```
1 CREATE PROCEDURE DajZakony
2 AS
3 SELECT IDZakon, Nazov FROM Zakony
4 GO
```

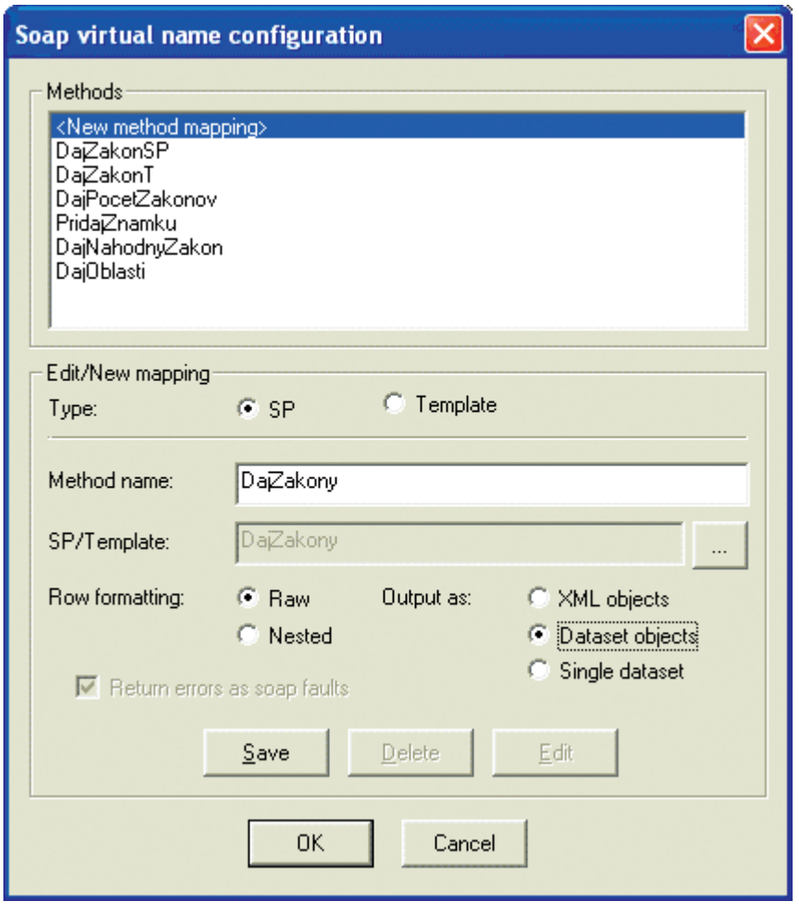
Nastavíme právo vykonávať túto procedúru používateľovi, ktorý bol definovaný pri konfigurácii virtuálneho adresára vaMuzaDB.

Mapovanie uloženej procedúry DajZakony do webovej služby ukazuje obr. 5.4. Všimnite si, že je nastavený výstup ako Dataset object.

Teraz nič nebráni tomu, aby sme novovytvorenú metódu webovej služby použili v aplikácii Muza4. Pretože webová služba SoapMuzaDB bola zmenená, v projekte Muza4 treba obnoviť príslušnú webovú referenciu. Z okna Solution Explorer vyvoláme kontextové menu pre vaMuzaDB a zvolíme Update Web Reference.

Do projektu zaraďme nové tlačidlo - ButDajZakony. Vytvoríme obsluhu udalosti stlačenia tohto tlačidla (viď tab. 5.4). Z webovej služby SoapMuzaDB získame pole objektov. Prvý z nich je typu DataSet. Odovzdáme ho do datagridu.

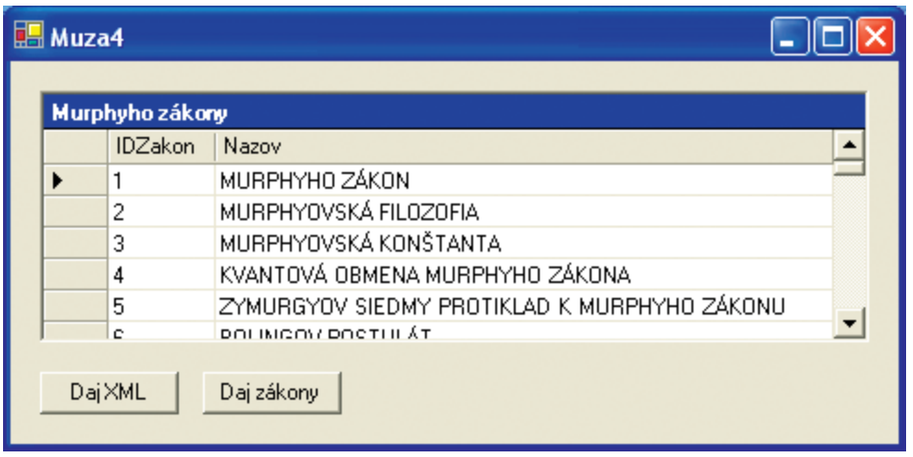
Poznamenajme, že pre zobrazenie zoznamu zákonov je použitý ten istý datagrid, ako pre zobrazenie jedného zákona. Výsledok získaný stlačením tlačidla ButDajZakony ukazuje obr. 5.5.



Obr. 5.4. Mapovanie uloženej procedúry do metódy webovej služby.

Tab. 5.4. Prevzatie objektu typu DataSet z webovej služby.

```
1 private void ButDajZakony_Click(object sender, System.EventArgs e)
2 { // IB **********************
3   SoapMuzaDB Muza = new SoapMuzaDB();
4   object[] Objekty = Muza.DajZakony();
5   dataGridMuza.DataSource = (DataSet)Objekty[0];
6   dataGridMuza.DataMember = "row";
7   dataGridMuza.CaptionText = "Murphyho zákony";
8 }
```



Obr. 5.5. Objekt typu DataSet získaný z webovej služby.

5.4. Murphyho zákon v objekte na strane servera

Môžu vzniknúť situácie, keď je rozumné, aby sme volaním webovej služby získali objekt a mali možnosť manipulovať s údajmi v podobe jeho členských premenných. Najjednoduchší spôsob, ako získať potrebné triedy, je využiť Xsd.exe. Tento nástroj je súčasťou .NET Framework SDK. Poskytuje viaceré služby, okrem iného aj možnosť vytvoriť potrebné triedy zo zadanej XML schémy. Predpis pre získanie triedy, je takýto:

xsd.exe <schema>.xsd /classes|dataset [/e:] [/l:] [/n:] [/o:] [/uri:]

<schema> je meno súboru XML schémy,

/classes - voľba prikazujúca vytvoriť triedy, skrátený zápis '/c',

/dataset - voľba prikazujúca vytvoriť DataSet, skrátený zápis '/d',

/element:<element> - element schémy, ktorý má byť spracovaný, skrátený zápis '/e:',

/language:<language> - použitý programovací jazyk: 'CS', 'VB', 'JS' , default je 'CS' (CSharp), skrátený zápis '/l:',

/namespace:<namespace> - priestor mien pre vytvorené súbory, skrátený zápis '/n:',

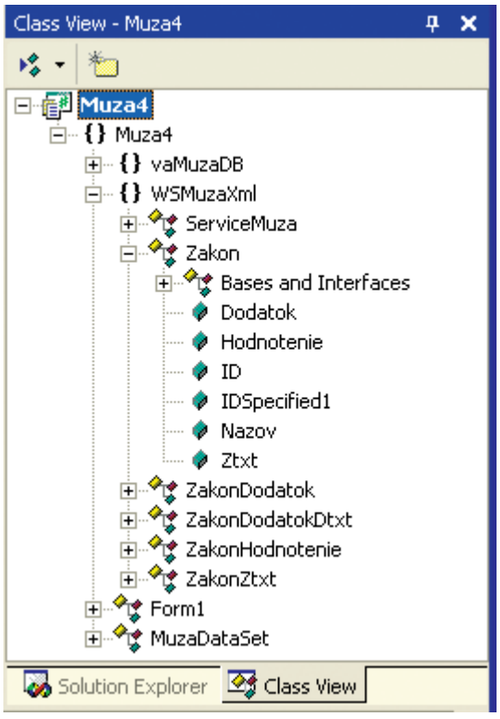
/out:<directoryName> - výstupný adresár pre vytvorené súbory, skrátený zápis '/o:',

/uri:<uri> - URI schémy, ktorá má byť spracovaná, skrátený zápis '/u:'.

Pre vytvorenie tried Murphyho zákona sme použili tieto voľby:

xsd.exe Muza4.xsd /classes /element:Zakon /namespace:Muza4Obj /language:CS /out:Xsd

Grafickú reprezentáciu použitej XML schémy Muza4.xsd vidieť na obr. 5.2. Výstup bol nasmerovaný do adresára Xsd. V ňom bol vytvorený súbor Muza4.cs. Premenovali sme ho na Muza4Obj.cs. Takto pripravený súbor sme zaradili do projektu webovej služby WSMuzaXml. Pohľad na vytvorené triedy zachytáva obr. 5.6.



Obr. 5.6. Triedy vytvorené pomocou Xsd.exe pre jeden Murphyho zákon.

Skutočnosť, že sme vytvorené triedy zaradili do projektu WSMuzaXML, naznačuje, že v tejto webovej službe vytvoríme objekt typu Zakon. Tento objekt bude návratovou hodnotou jednej z metód webovej služby. Pripomeňme, že projekt sme začali budovať v časti 3.5. Údaje sme v ňom čerpali z XML súboru, ktorý obsahoval Murphyho zákony. Namiesto toho, aby sme pokračovali v práci s XML súborom, využijeme vo webovej službe inú webovú službu - službu SoapMuzaDB. Okrem generovania objektu tak budeme mať príležitosť ukázať aj „zreťazenie“ webových služieb. Pridajme teda vo webovej službe WSMuzaDB webovú referenciu na službu SoapMuzaDB. Získame z nej náhodne vybraný Murphyho zákon. Výsledok zabalíme do objektu a ten bude návratovou hodnotou. Naznačené riešenie je konkretizované v tab. 5.5.

Tab. 5.5. Metóda webovej služby WSMuzaXml vracajúca objekt.

```
1  [WebMethod
2    (Description="Vráti náhodne vybraný Murphyho zákon ako objekt")]
3  [SoapRpcMethod]
4  public Zakon DajNahodnyZakonObj()
5  { // IB ***********************
6    SoapMuzaDB Muza = new SoapMuzaDB();
7    object[] PoleObjektov=Muza.DajNahodnyZakon();
8    XmlElement root= (XmlElement)PoleObjektov[0];
9
10   Zakon Z = new Zakon();
11
12   IEnumerator enumRoot = root.GetEnumerator();
13   while (enumRoot.MoveNext())
14   { // Zoberiem aktuálny prvok - node a spracujem ho
15     XmlNode node = (XmlNode) enumRoot.Current;
16     if (node.Name == "Zakon")
17     {
18       Z.Nazov = node.Attributes["Nazov"].Value;
19       Z.ID     = Int32.Parse(node.Attributes["ID"].Value);
20       Z.IDSpecified=true;
21       int iZtxt=0;
22       int iDodatok=0;
23
```

```
24        IEnumerator enumZakon = node.GetEnumerator();
25        iZtxt = Pocet (enumZakon, "Ztxt");
26        if (iZtxt>0)     Z.Ztxt     = new ZakonZtxt[iZtxt];
27        iDodatok = Pocet (enumZakon, "Dodatok");
28        if (iDodatok>0) Z.Dodatok = new ZakonDodatok[iDodatok];
29
30        iZtxt=iDodatok=0;
31        enumZakon.Reset();
32        while (enumZakon.MoveNext())
33        {
34          XmlNode nodeZakona = (XmlNode) enumZakon.Current;
35          switch (nodeZakona.Name)
36          {
37            case "Ztxt":
38            {
39              ZakonZtxt Ztxt = new ZakonZtxt();
40              Ztxt.Value=nodeZakona.InnerText;
41              Z.Ztxt.SetValue(Ztxt,iZtxt);
42              iZtxt++;
43            }
44            break;
45
46            case "Dodatok":
47            {
48              ZakonDodatok Dodatok = new ZakonDodatok();
49              Dodatok.Nazov = nodeZakona.Attributes["Nazov"].Value;
50              Z.Dodatok.SetValue(Dodatok,iDodatok);
51              int iDtxt=0;
52              IEnumerator enumDodatok = nodeZakona.GetEnumerator();
53              iDtxt = Pocet (enumDodatok, "Dtxt");
54              if (iDtxt>0)
55                Z.Dodatok[iDodatok].Dtxt = new ZakonDodatokDtxt[iDtxt];
56              iDtxt=0;
57              enumDodatok.Reset();
58              while (enumDodatok.MoveNext())
59              { XmlNode NodeDtxt = (XmlNode) enumDodatok.Current;
60                if (NodeDtxt.Name == "Dtxt")
61                {
62                  ZakonDodatokDtxt Dtxt = new ZakonDodatokDtxt();
63                  Dtxt.Value=NodeDtxt.InnerText;
64                  Z.Dodatok[iDodatok].Dtxt.SetValue(Dtxt,iDtxt);
65                  iDtxt++;
66                }
67              }
68              iDodatok++;
69            }
70            break;
71
72            case "Hodnotenie":
73            {
74              ZakonHodnotenie H = new ZakonHodnotenie();
75              H.Znamka =
76                Double.Parse(nodeZakona.Attributes["Znamka"].Value);
77              H.ZnamkaSpecified=true;
78              H.Pocet =
79                Int32.Parse(nodeZakona.Attributes["Pocet"].Value);
80              H.PocetSpecified=true;
81              H.Cas = DateTime.Parse(nodeZakona.Attributes["Cas"].Value);
82              H.CasSpecified=true;
```

```
 83              Z.Hodnotenie = new ZakonHodnotenie[1];
 84              Z.Hodnotenie.SetValue(H,0);
 85            }
 86            break;
 87          }
 88        }
 89      }
 90    }
 91    return Z;
 92  }
 93
 94  private int Pocet (IEnumerator ienum, string Znacka)
 95  { // IB *********
 96    int nVyskytov=0;
 97    ienum.Reset();
 98    while (ienum.MoveNext())
 99    {
100      XmlNode node = (XmlNode) ienum.Current;
101      if (node.Name == Znacka)    nVyskytov++;
102    }
103    return nVyskytov;
104  }
```

Vidíme, že vytvorenie objektu typu Zakon nie je jednoduché. Je to dané tým, že ani vlastná XML schéma takého zákona nie je jednoduchá. Útechou nech nám je, že ak vynaložíme námahu pri tvorbe objektu vo webovej službe, uľahčíme spracovanie údajov na strane klienta. Poďme sa pozrieť na vytvorený kód.

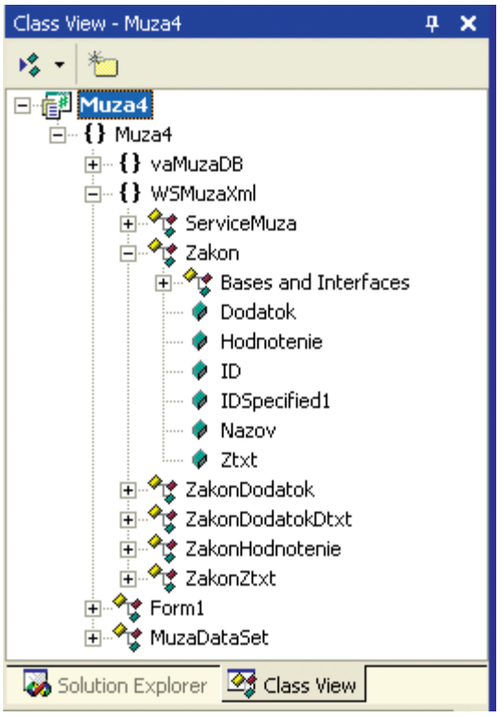
Atribút WebMethod naznačuje, že nasledujúca metóda bude metódou webovej služby. Tak ako vo všetkých doterajších metódach webovej služby, aj tu sme nastavili vlastnosť (property) Description. Nastavenou hodnotou definujeme opis webovej služby. Poznamenajme, že atribút WebMethod má aj ďalšie vlastnosti, ktorými sa dá riadiť:

- bafrovanie - vlastnosť BufferResponse,
- keš - vlastnosť CacheDuration,
- použitie objektu HttpSessionState z rodičovskej triedy - vlastnosť EnableSession,
- podiel na transakcii - vlastnosť TransactionOption,
- pomenovanie metód - vlastnosť MessageName.

Atribút SoapRpcMethod (riadok 3 tab 5.5) určuje, že v protokole SOAP budú očakávané správy pre podporu RPC (Remote Procedure Call). V riadku 4 si treba všimnúť, že návratovým typom metódy je Zakon. Z webovej služby SoapMuzaDB získame náhodne vybraný Murphyho zákon vo forme XmlElementu. Našou úlohou je vybrať z neho údaje a zapísať ich do členských premenných objektu Z typu Zakon. Situáciu sťažuje to, že v triede Zakon sú niektoré členské premenné polia. Pred ich vytvorením potrebujeme poznať počet položiek. Využívame k tomu pomocnú metódu - Pocet (riadky 94 až 104). Vlastné naplnenie objektu Z údajmi sa uskutočňuje v cykle (začína riadkom 13). Prechádzame v ňom po prvkoch spracovávaného Murphyho zákona a údaje zapisujeme do členských premenných objektu Z.

5.5. Murphyho zákon v objekte na strane klienta

V predošlej časti sme ukázali rozšírenie webovej služby WSMuzaXML o novú metódu. Overíme ju v aplikácii Muza4. Prvým krokom je obnovenie referencie tejto webovej služby. Popri triede ServiceMuza, ktorá v našom projekte reprezentuje webovú službu WSMuzaXML, získame aj triedy umožňujúce pracovať s Murphyho zákonom ako s objektom. Ukazuje to obr. 5.7.



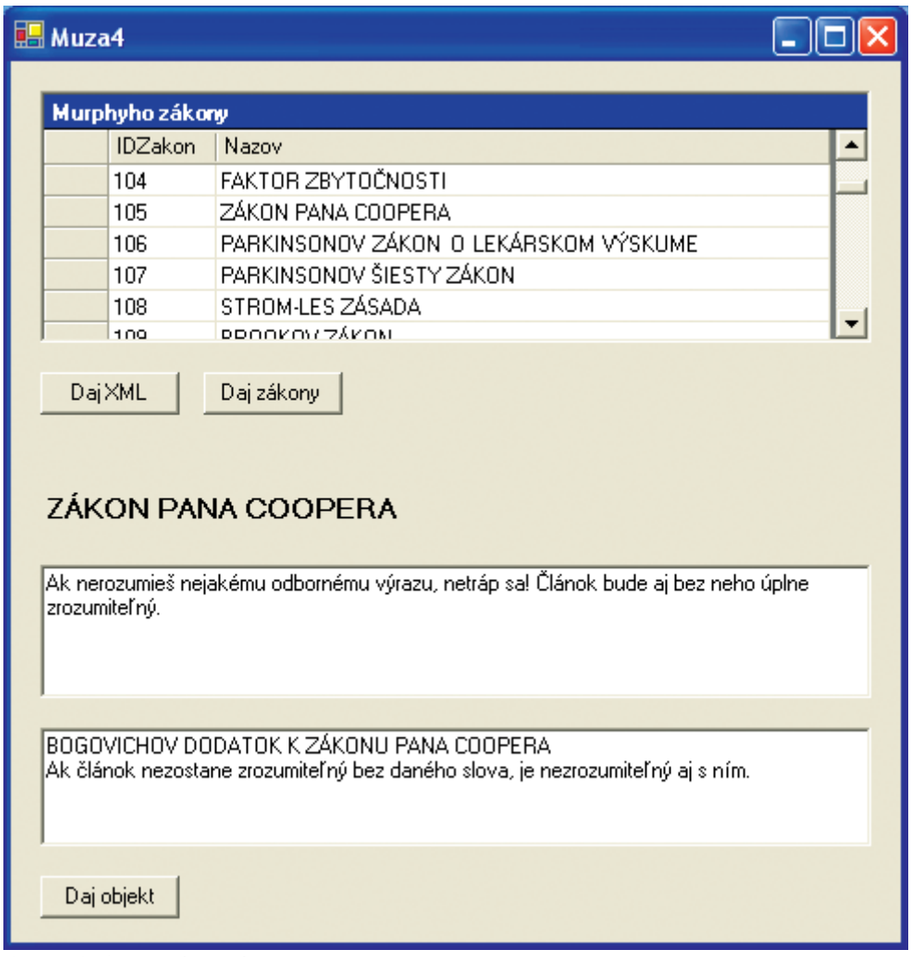
Obr. 5.7. Triedy získané obnovou webovej referencie.

Webovú službu teraz môžeme použiť pre získanie objektu typu Zakon a priamo pracovať s jeho členskými premennými. Pre overenie tejto možnosti zaradíme do formulára projektu tlačidlo ButDajObjekt. Obsluha stlačenia tohto tlačidla je v tab. 5.6. Pre výpis získaných údajov sme použili LabeNazov, TextBoxZtxt a TextBoxDtxt.

Tab. 5.6. Získanie a spracovanie objektu na strane klienta.

```
 1 private void BudDajObjekt_Click(object sender, System.EventArgs e)
 2 { // IB ***********************
 3   ServiceMuza Muza = new ServiceMuza();
 4   Zakon Z = new Zakon();
 5   Z = Muza.DajNahodnyZakonObj();
 6   LabelNazov.Text = Z.Nazov;
 7   TextBoxZtxt.Text="";
 8   if (Z.Ztxt != null) foreach  (ZakonZtxt Ztxt in Z.Ztxt)
 9     TextBoxZtxt.Text+=Ztxt.Value+"\r\n";
10   TextBoxDtxt.Text="";
11   if (Z.Dodatok != null) foreach (ZakonDodatok Dodatok in Z.Dodatok)
12   { TextBoxDtxt.Text+=Dodatok.Nazov+"\r\n";
13     if (Dodatok.Dtxt != null)
14       foreach  (ZakonDodatokDtxt Dtxt in Dodatok.Dtxt)
15         TextBoxDtxt.Text+=Dtxt.Value+"\r\n";
16     TextBoxDtxt.Text+="\r\n";
17   }
18 }
```

Objekt Z typu Zakon je naplnený volaním webovej služby (riadok 5). Príklady priamej manipulácie s členskými premennými tohto objektu sú v nasledujúcich riadkoch kódu. Výsledok vidíme na obr. 5.8.



Obr. 5.8. Udaje získané prenosom objektu.

Pre zaujímavosť je možné porovnať kód v tab. 5.6. s kódom v tab. 3.4. V procedúre ZobrazZakon sme tam riešili podobnú úlohu. Údaje Murphyho zákona sme tam získavali z textového reťazca vo forme XML a manipulovali sme s ním prostredníctvom triedy XmlDocument.

5.6. XML serializácia

V predošlých dvoch statiach sme mali možnosť vidieť, že objekt vytvorený na strane servera bol prenesený cez sieťové prostredie. Na strane klienta sme mali možnosť manipulovať s členskými premennými tohto objektu. V pozadí týchto možností je XLM serializácia a deserializácia. Serializácia je proces, ktorý na strane servera umožňuje konvertovať objekty do tvaru ľahko prenesiteľného transportnými protokolmi ku klientovi. Naproti tomu deserializácia je proces, ktorý z prenesených údajov na strane klienta umožňuje vytvoriť objekty. Z opísaného postupu tvorby webovej služby a klientskej aplikácie vidieť, že programátor má k dispozícii automatizačné prostriedky, ktoré za neho vygenerujú príslušný kód. Pozreli sme sa naň iba cez pohľady na triedy - obr. 5.6 na strabe servera a obr. 5.7 na strane klienta. Preskúmajme vygenerovaný kód.

V tab 5.7 je zdrojový kód tried, ktorý sme získali pomocou Xsd.exe z .NET Framework SDK (viď opis v 5.4). Použili sme ho na strane servera.

Tab. 5.7. Triedy opisujúce Murphyho zákon na strane servera.

```
//------------------------------------------------------------------
// <autogenerated>
//      This code was generated by a tool.
//      Runtime Version: 1.0.3705.0
//
//      Changes to this file may cause incorrect behavior and will be
lost if
//      the code is regenerated.
// </autogenerated>
//------------------------------------------------------------------

//
// This source code was auto-generated by xsd, Version=1.0.3705.0.
//
namespace Muza4Obj {
    using System.Xml.Serialization;


    /// <remarks/>

[System.Xml.Serialization.XmlTypeAttribute(Namespace="http://buransky.
sk/Muza4.xsd")]

[System.Xml.Serialization.XmlRootAttribute(Namespace="http://buransky.
sk/Muza4.xsd", IsNullable=false)]
    public class Zakon {

        /// <remarks/>
        [System.Xml.Serialization.XmlElementAttribute("Ztxt",
IsNullable=true)]
        public ZakonZtxt[] Ztxt;

        /// <remarks/>
        [System.Xml.Serialization.XmlElementAttribute("Dodatok")]
        public ZakonDodatok[] Dodatok;

        /// <remarks/>
        [System.Xml.Serialization.XmlElementAttribute("Hodnotenie")]
        public ZakonHodnotenie[] Hodnotenie;

        /// <remarks/>

[System.Xml.Serialization.XmlAttributeAttribute(Form=System.Xml.Schema.XmlSchemaForm.Unquali
fied)]
        public string Nazov;

        /// <remarks/>

[System.Xml.Serialization.XmlAttributeAttribute(Form=System.Xml.Schema.XmlSchemaForm.Unquali
fied)]
        public int ID;

        /// <remarks/>
        [System.Xml.Serialization.XmlIgnoreAttribute()]
        public bool IDSpecified;
    }

```

```
58      /// <remarks/>
59
60  [System.Xml.Serialization.XmlTypeAttribute(Namespace="http://buransky.
61  sk/Muza4.xsd")]
62      public class ZakonZtxt {
63
64          /// <remarks/>
65          [System.Xml.Serialization.XmlTextAttribute()]
66          public string Value;
67      }
68
69      /// <remarks/>
70
71  [System.Xml.Serialization.XmlTypeAttribute(Namespace="http://buransky.
72  sk/Muza4.xsd")]
73      public class ZakonHodnotenie {
74
75          /// <remarks/>
76
77  [System.Xml.Serialization.XmlAttributeAttribute(Form=System.Xml.Schema.XmlSchemaForm.Unquali
78  fied)]
79          public System.Double Znamka;
80
81          /// <remarks/>
82          [System.Xml.Serialization.XmlIgnoreAttribute()]
83          public bool ZnamkaSpecified;
84
85          /// <remarks/>
86
87  [System.Xml.Serialization.XmlAttributeAttribute(Form=System.Xml.Schema.XmlSchemaForm.Unquali
88  fied)]
89          public int Pocet;
90
91          /// <remarks/>
92          [System.Xml.Serialization.XmlIgnoreAttribute()]
93          public bool PocetSpecified;
94
95          /// <remarks/>
96
97  [System.Xml.Serialization.XmlAttributeAttribute(Form=System.Xml.Schema.XmlSchemaForm.Unquali
98  fied)]
99          public System.DateTime Cas;
100
101         /// <remarks/>
102         [System.Xml.Serialization.XmlIgnoreAttribute()]
103         public bool CasSpecified;
104     }
105
106     /// <remarks/>
107
108 [System.Xml.Serialization.XmlTypeAttribute(Namespace="http://buransky.
109 sk/Muza4.xsd")]
110     public class ZakonDodatokDtxt {
111
112         /// <remarks/>
113         [System.Xml.Serialization.XmlTextAttribute()]
114         public string Value;
115     }
```

```
116
117     /// <remarks/>
118
119 [System.Xml.Serialization.XmlTypeAttribute(Namespace="http://buransky.
120 sk/Muza4.xsd")]
121     public class ZakonDodatok {
122
123         /// <remarks/>
124         [System.Xml.Serialization.XmlElementAttribute("Dtxt",
125 IsNullable=true)]
126         public ZakonDodatokDtxt[] Dtxt;
127
128         /// <remarks/>
129
130 [System.Xml.Serialization.XmlAttributeAttribute(Form=System.Xml.Schema.XmlSchemaForm.Unquali
131 fied)]
132         public string Nazov;
133     }
134 }
```

Pri skúmaní vytvoreného kódu v tab. 5.7 našej pozornosti neuniknú atribúty serializácie. Tak napr. v riadkoch 21, 22 je použitý XmlTypeAttribute, v ktorom je definovaný priestor mien. V riadkoch 24, 25 je atribútom XmlRootAttribute - označuje triedu, ktorá reprezentuje koreňový prvok XML dokumentu. Oba tieto atribúty sú pred triedou Zakon. Členské premenné triedy Zaokon sú označené atribútom XmlElementAttribute, alebo XmlAttributeAttribute. Označuje sa nimi, či príslušná premenná bude serializovaná ako XML element, alebo ako atribút - viď napr. riadky 29,30 resp. 43, 44. Pozornému čitateľoli zrejme neuniknú ani ďalšie atribúty.

Pozrime sa na triedy, ktoré sme v stati 5.5 na strane klienta získali pridaním webovej referencie na webovú službu. Sú v tab 5.8.

Tab. 5.8. Triedy získané pridaním webovej referencie na strane klienta.

```
 1 //------------------------------------------------------------------
 2 // <autogenerated>
 3 //      This code was generated by a tool.
 4 //      Runtime Version: 1.0.3705.0
 5 //
 6 //      Changes to this file may cause incorrect behavior and will be
 7 lost if
 8 //      the code is regenerated.
 9 // </autogenerated>
10 //------------------------------------------------------------------
11
12 //
13 // This source code was auto-generated by Microsoft.VSDesigner,
14 Version 1.0.3705.0.
15 //
16 namespace Muza4.WSMuzaXml {
17     using System.Diagnostics;
18     using System.Xml.Serialization;
19     using System;
20     using System.Web.Services.Protocols;
21     using System.ComponentModel;
22     using System.Web.Services;
23
24
25     /// <remarks/>
26     [System.Diagnostics.DebuggerStepThroughAttribute()]
27     [System.ComponentModel.DesignerCategoryAttribute("code")]
```

```
28
29 [System.Web.Services.WebServiceBindingAttribute(Name="ServiceMuzaSoap",
30 Namespace="http://buransky.sk/WSMuzaXml/")]
31
32 [System.Xml.Serialization.SoapIncludeAttribute(typeof(ZakonHodnotenie))]
33
34
35 [System.Xml.Serialization.SoapIncludeAttribute(typeof(ZakonDodatokDtxt))]
36
37
38 [System.Xml.Serialization.SoapIncludeAttribute(typeof(ZakonDodatok))]
39 [System.Xml.Serialization.SoapIncludeAttribute(typeof(ZakonZtxt))]
40     public class ServiceMuza :
41 System.Web.Services.Protocols.SoapHttpClientProtocol {
42
43         /// <remarks/>
44         public ServiceMuza() {
45             this.Url = "http://localhost/WSMuzaXml/ServiceMuza.asmx";
46         }
47
48         /// <remarks/>
49
50 [System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://buransky.
51 sk/WSMuzaXml/DajPocetZakonov", RequestNamespace="http://buransky.sk/WSMuzaXml/",
52 ResponseNamespace="http://buransky.sk/WSMuzaXml/",
53 Use=System.Web.Services.Description.SoapBindingUse.Literal,
54 ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]
55
56
57         public int DajPocetZakonov() {
58             object[] results = this.Invoke("DajPocetZakonov", new object[0]);
59
60             return ((int)(results[0]));
61         }
62
63         /// <remarks/>
64         public System.IAsyncResult BeginDajPocetZakonov(System.AsyncCallback callback,
65 object asyncState) {
66
67             return this.BeginInvoke("DajPocetZakonov", new object[0], callback,
68 asyncState);
69         }
70
71         /// <remarks/>
72         public int EndDajPocetZakonov(System.IAsyncResult asyncResult) {
73
74             object[] results = this.EndInvoke(asyncResult);
75             return ((int)(results[0]));
76         }
77
78         /// <remarks/>
79      ... tu sú ďalšie metódy webovej služby ...
80
81     /// <remarks/>
82     [System.Xml.Serialization.SoapTypeAttribute("Zakon",
83 "http://buransky.sk/WSMuzaXml/encodedTypes")]
84     public class Zakon {
85
```

```
 86          /// <remarks/>
 87          public ZakonZtxt[] Ztxt;
 88
 89          /// <remarks/>
 90          public ZakonDodatok[] Dodatok;
 91
 92          /// <remarks/>
 93          public ZakonHodnotenie[] Hodnotenie;
 94
 95          /// <remarks/>
 96          public string Nazov;
 97
 98          /// <remarks/>
 99          public int ID;
100
101          /// <remarks/>
102          [System.Xml.Serialization.SoapElementAttribute("IDSpecified")]
103          public bool IDSpecified1;
104      }
105
106      /// <remarks/>
107      [System.Xml.Serialization.SoapTypeAttribute("ZakonZtxt",
108  "http://buransky.sk/WSMuzaXml/encodedTypes")]
109      public class ZakonZtxt {
110
111          /// <remarks/>
112          public string Value;
113      }
114
115      /// <remarks/>
116      [System.Xml.Serialization.SoapTypeAttribute("ZakonHodnotenie",
117  "http://buransky.sk/WSMuzaXml/encodedTypes")]
118      public class ZakonHodnotenie {
119
120          /// <remarks/>
121          public System.Double Znamka;
122
123          /// <remarks/>
124
125  [System.Xml.Serialization.SoapElementAttribute("ZnamkaSpecified")]
126          public bool ZnamkaSpecified1;
127
128          /// <remarks/>
129          public int Pocet;
130
131          /// <remarks/>
132
133  [System.Xml.Serialization.SoapElementAttribute("PocetSpecified")]
134          public bool PocetSpecified1;
135
136          /// <remarks/>
137          public System.DateTime Cas;
138
139          /// <remarks/>
140
141  [System.Xml.Serialization.SoapElementAttribute("CasSpecified")]
142          public bool CasSpecified1;
143      }
```

```
144
145     /// <remarks/>
146     [System.Xml.Serialization.SoapTypeAttribute("ZakonDodatokDtxt",
147 "http://buransky.sk/WSMuzaXml/encodedTypes")]
148     public class ZakonDodatokDtxt {
149
150         /// <remarks/>
151         public string Value;
152     }
153
154     /// <remarks/>
155     [System.Xml.Serialization.SoapTypeAttribute("ZakonDodatok",
156 "http://buransky.sk/WSMuzaXml/encodedTypes")]
157     public class ZakonDodatok {
158
159         /// <remarks/>
160         public ZakonDodatokDtxt[] Dtxt;
161
162         /// <remarks/>
163         public string Nazov;
164     }
165 }
```

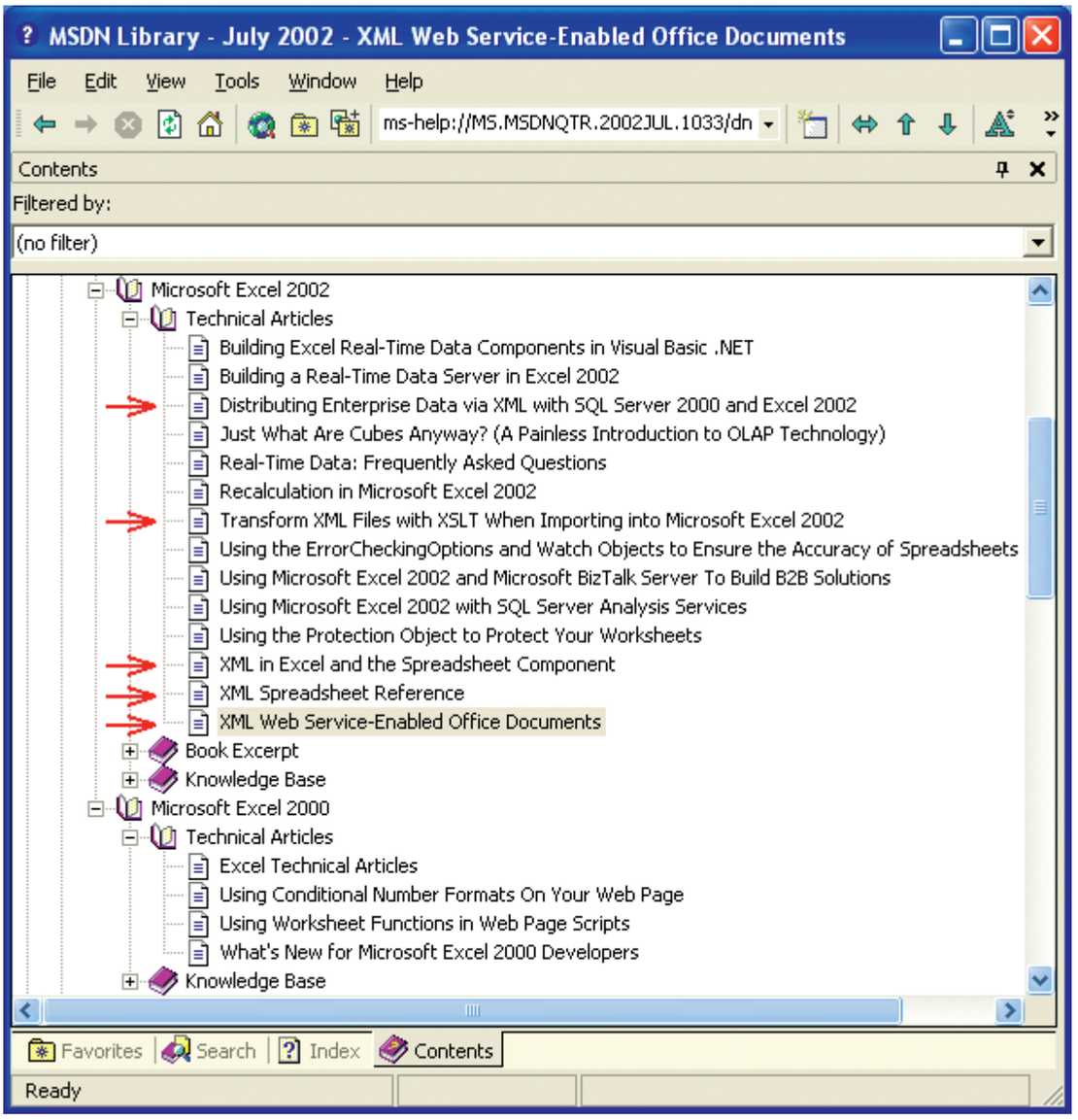
Kód triedy ServiceMuza, ktorá v klientskej aplikácii zastupuje webovú službu, začína riadkom 40. Pred ním je množina atribútov, ktoré upresňujú serializáciu. Pre skrátenie tabuľky sme ponechali kód potrebný pre volanie iba jednej metódy webovej služby - DajPocetZakonov (riadky 50 až 76). Riadkom 82 začína kód tried, ktoré opisujú objekty Murphyho zákona. Všimnite si atribút SoapTypeAttribute, ktorý predchádza týmto triedam, prípadne aj SoapElementAttribute (napr. v riadku 102).

Vysvetlenie všetkých atribútov seializácie presahuje zámer tejto publikácie, a tak čitateľom, ktorí budú mať záujem preniknúť hlbšie do problematiky serializácie, môžeme odporučiť začať so štúdiom state „Introducing XML Serialization“ v .NET Framework Developer's Guide.

6. XML a MS Office

Motto: ZÁSADA PLÁNOVACEJ NESCHOPNOSTI
Každá zmena spočiatku vyzerá strašne.

XML a súvisiace technológie prenikajú aj do takých produktov, ako sú kancelárske balíky. Najnovšie zmeny v balíku MS Office sa týkajú predovšetkým možnosti použitia XML a webových služieb. Porovnajme obsah článkov v MSDN Library, ktoré sa vzťahujú k Microsoft Excel 2000 a 2002. Päť článkov, v ktorých sa priamo vyskytuje skratka XML, je na obrázku 6.1 označených červenou šípkou.

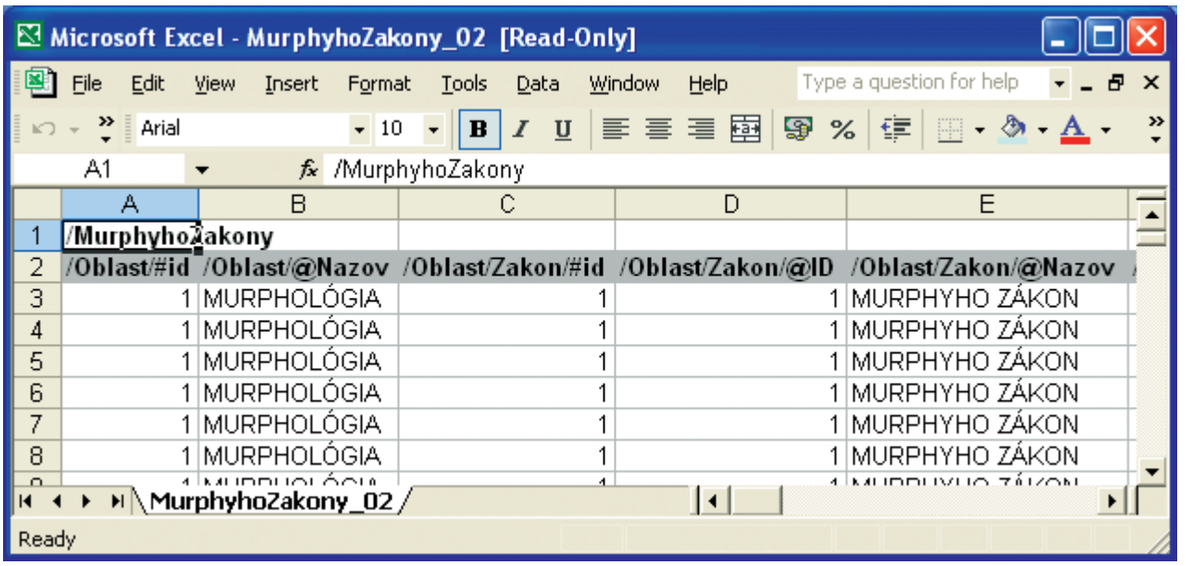


Obr. 6.1. XML zasiahol aj Microsoft Excel.

Preto sa pokúsime nahliadnuť do možností, ktoré pre XML a webové služby ponúka Microsoft Excel.

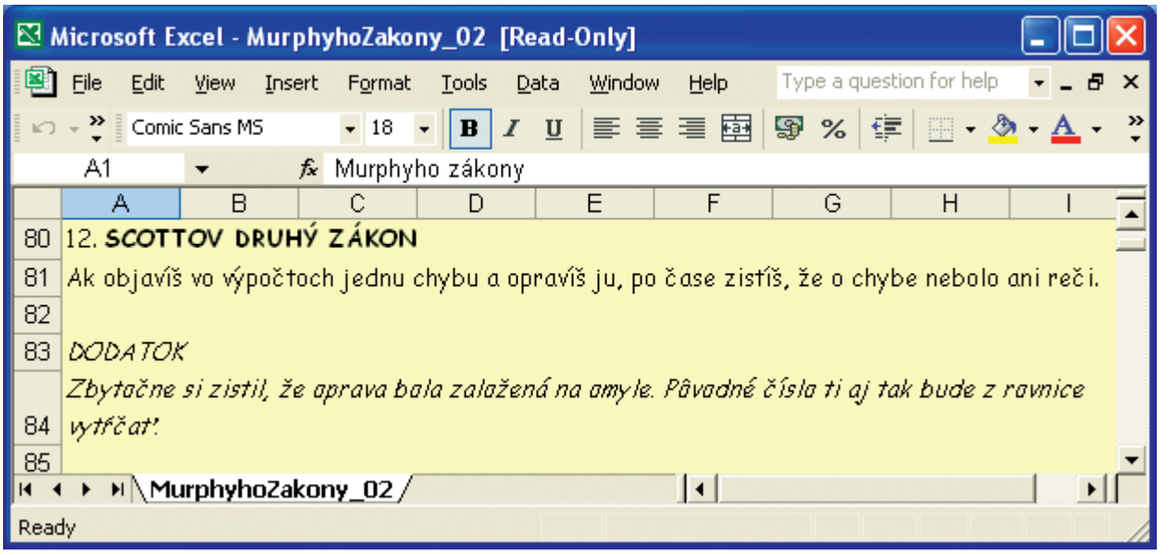
6.1. Použite XML súborov v Exceli

Microsoft Excel dokáže načítať XML súbor. Na obr. 6.2 je výrez zobrazenia XML súboru Murphyho zákonov.



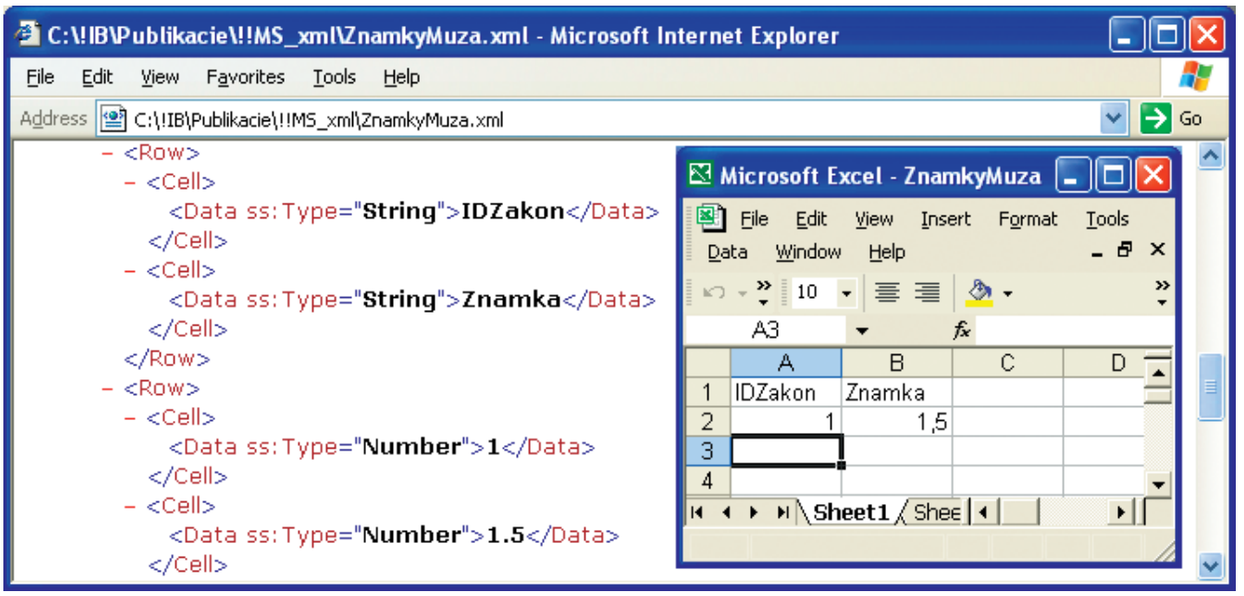
Obr. 6.2. Murphyho zákony v Exceli.

Microsoft Excel dokáže zobraziť aj XML súbor, v ktorom je odkaz na súbor XSL transformácie. Ukazuje to obr. 6.3.



Obr. 6.3. Murphyho zákony v Exceli s uplatnením XSL transformácie.

Dokumenty vytvorené v Exceli možu byť zapísané v XML formáte. Ukazuje to obr. 6.4, kde je dokument v Exceli a výsledný XML dokument zobrazený v Internet exploreri.



Obr. 6.4. Výstup z Excelu vo forme XML dokumnetu.

Uvedené ukázky svedčia o tom, že Microsoft Excel je vybavený „schopnosťami“ práce s XML dokumentami. Tam sa však jeho možnosti nekončia. Dajú sa v ňom využiť aj webové služby.

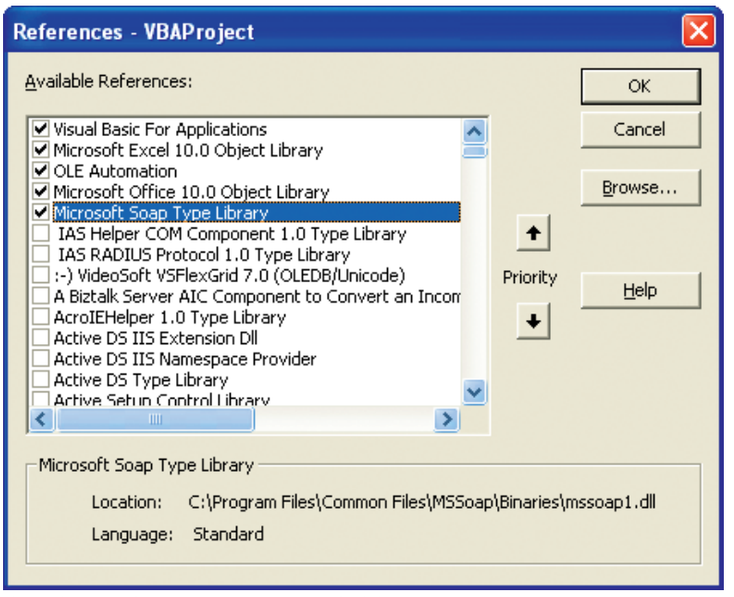
6.2. Využitie webových služieb v Exceli

Využitie webových služieb v kancelárskom balíku Microsoft Office sa dá dosiahnuť prostriedkami VBA (Visual Basic for Applications). K tomu je potrebné:

- Microsoft Soap Type Library (mssoap1.dll). Je súčasťou Microsoft Windows XP, ale dá sa získať aj z iných zdrojov, napr. Microsoft SOAP Toolkit (v súčasnosti je k dispozícii verzia 3.0) alebo z Office XP Web Services Toolkit.
- Microsoft XML Parser (v súčasnosti je k dispozícii už štvrtá verzia msxml4.dll).

Tak ako doteraz, aj naďalej sa budeme „zabávať“ na účet Murphyho zákonov. Budeme využívať webovú službu WSMuzaXml, ktorú sme začali tvoriť v časti 3.5 a rozširovali v 4.8. Pre ukážku použitia webovej služby v Microsoft Excel je táto služba rozšírená o metódu DajNazovZakona. Pre zadané číslo (identifikátor) zákona vydá jeho názov ako textový reťazec. Pri poznaní opísaných postupov v predošlých častiach je vytvorenie novej metódy webovej služby vecou „rutiny“. Našou úlohou bude vytvoriť používateľskú funkciu, ktorá pre zadané číslo zistí názov Murphyho zákona - volaním metódy webovej služby.

Otvorme v Microsoft Excel nový zošit - nazvime ho Muza. Z menu Tools - Macro otvorme editor jazyka Visual Basic. V ňom pridajme referenciu na Microsoft Soap Type Library, ako ukazuje obr. 6.5.



Obr. 6.5. Referencia na Microsoft SOAP Type Library vo VBA.

Doplňme do projektu nový modul a definujme v ňom funkciu, ktorej zdrojový kód je v tab. 6.1.

Tab. 6.1. Používateľská funkcia NazovMuza1 pre získanie názvu Murphyho zákona.

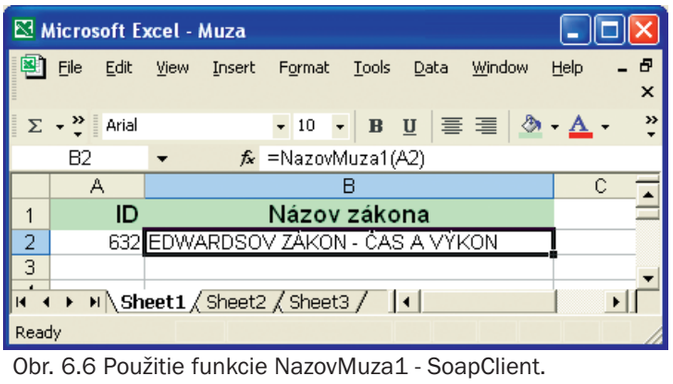
```
1 Public Function NazovMuza1(ByVal ID As Long) As String
2   Application.Volatile False
3   Set C = New SoapClient
4   C.MSSoapInit "http://localhost/WSMuzaXml/ServiceMuza.asmx?WSDL"
5   NazovMuza1 = C.DajNazovZakona(ID)
6   Set C = Nothing
7 End Function
```

Argumentom funkcie NazovMuza1 je identifikátor zákona, ktorého názov chceme získať. Funkcia vracia textový reťazec. Zápisom

```
Application.Volatile False
```

v riadku 2 zabezpečíme, že funkcia bude volaná iba pri zmene vstupných údajov. Referencia na Microsoft SOAP Type Library umožňuje vytvoriť objekt typu SoapClient - riadok 3. V riadku 4 je inicializácia objektu SOAP klient. Pre inicializáciu odovzdávame textový reťazec - adresu, na ktorej je opis webovej služby. Potom nasleduje volanie metódy webovej služby (riadok 5) .

Použitie vytvorenej používateľskej funkcie ukazuje obr. 6.6.



Obr. 6.6 Použitie funkcie NazovMuza1 - SoapClient.

Poznamenajme, že uvedený postup sa dá použiť nielen v Office XP ale aj Office 2000. V prostredí Windows XP je Microsoft Soap Type Library (mssoap1.dll) priamo k dispozícii. Je však rozumné využiť Microsoft SOAP Toolkit 3.0. Dá sa získať na adrese:

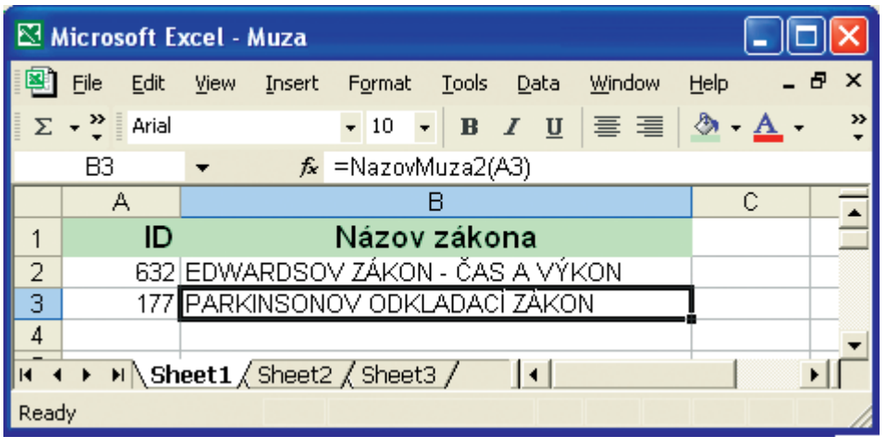
http://www.microsoft.com/downloads

Po nainštalovaní SOAP Toolkit 3.0 popri mssoap1.dll získame možnosť využiť Microsoft Soap Type Library v3.0 (mssoap30.dll). Pridajme do nášho projektu VBA referenciu na tento prostriedok. Potom môžeme vytvoriť používateľskú funkciu, v ktorej budeme namiesto objektu typu SoapClient využívať objekt typu SoapClient30. Ukazuje to tab. 6.2.

Tab. 6.2. Používateľská funkcia NazovMuza2 pre získanie názvu Murphyho zákona.

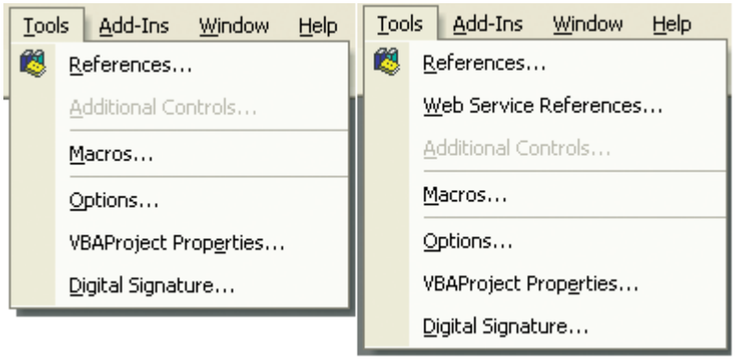
```
1  Public Function NazovMuza2(ByVal ID As Long) As String
2    Application.Volatile False
3    Set C = CreateObject("MSSOAP.SoapClient30")
4    C.MSSoapInit "http://localhost/WSMuzaXml/ServiceMuza.asmx?WSDL"
5    NazovMuza2 = C.DajNazovZakona(ID)
6    Set C = Nothing
7  End Function
```

Použitie funkcie NazovMuza2 je na obr. 6.7.



Obr. 6.7. Použitie funkcie NazovMuza2 - SoapClient30.

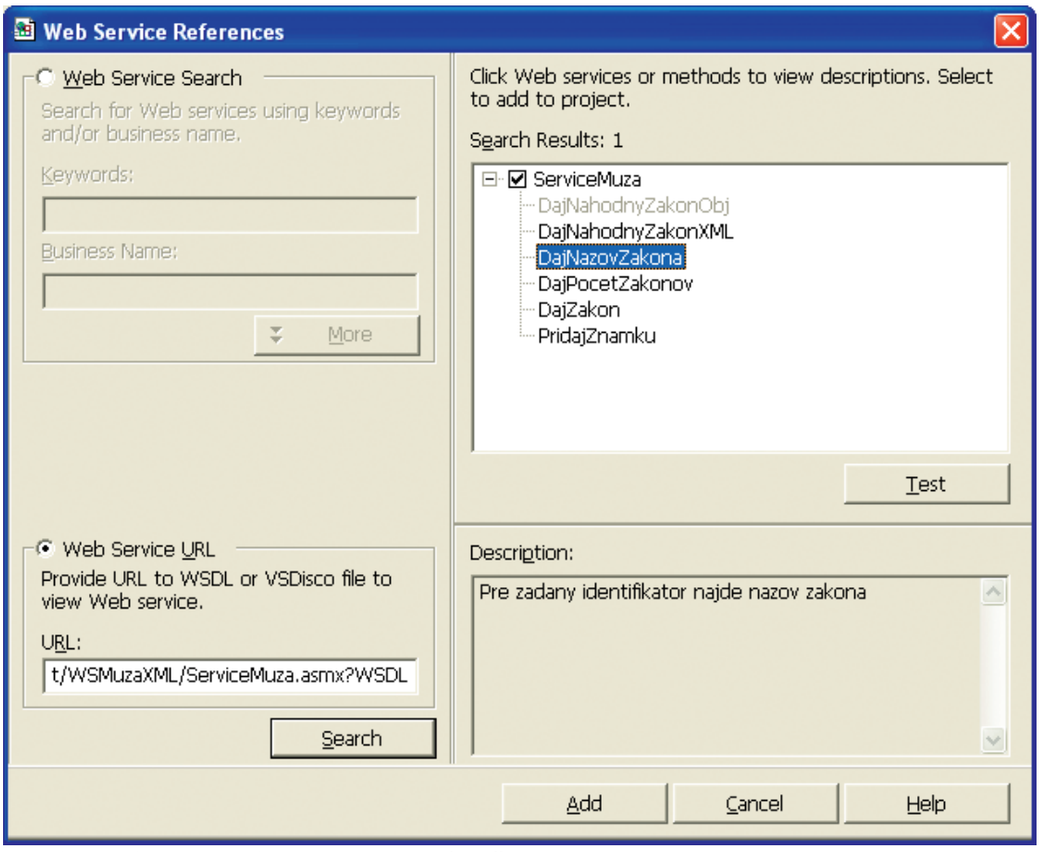
V oboch používateľských funkciách - NazovMuza1, NazovMuza2 - sme vytvárali webovú referenciu vytvorením klienta SOAP, jeho inicializáciou a volaním metódy webovej služby. Túto činnosť nám „spríjemní“ Office XP Web Services Toolkit, ktorý je voľne k dispozícii na *http://msdn.microsoft.com/downloads*. Umožní nám pohodlnejšie vytvárať referencie na webové služby. Po jeho nainštalovaní v menu nástojov editora VBA získame položku **Web Service References** ..., ako ukazuje obr. 6.8.



Obr. 6.8. Nástroje VBA pred a po nainštalovaní Office XP Web Services Toolkit.

Treba poznamenať, že aj napriek zvýrazneniu XP v názve produktu, je možné jeho použitie aj pre Office 2000.

Získaný nástroj pre vytváranie referencií na webové služby ukazuje obr. 6.9.



Obr. 6.9. Definovanie referencie na webovú službu.

Ak sú k dispozícii služby UDDI servera, webové služby je možné vyhľadávať podľa zadaných kľúčových slov a obchodných názvov. Webovú službu je možné lokalizovať aj zadaním URL na jej opis. Práve takú situáciu zachytáva obr. 6.9 - po zadaní URL a stlačení tlačidla Search. V okne Search Results je zobrazená informácia o vyhľadanej webovej službe a jej metódach. Po označení služby a stlačení tlačidla Add bude webová služba zahrnutá do projektu. Vygenerovaná trieda zástupcu zvolenej webovej služby vo VBA projekte má názov clsws_ServiceMuza. Je uvedená v tab. 6.3.

Tab. 6.3. Trieda vygenerovaná nástrojom Web Service References.

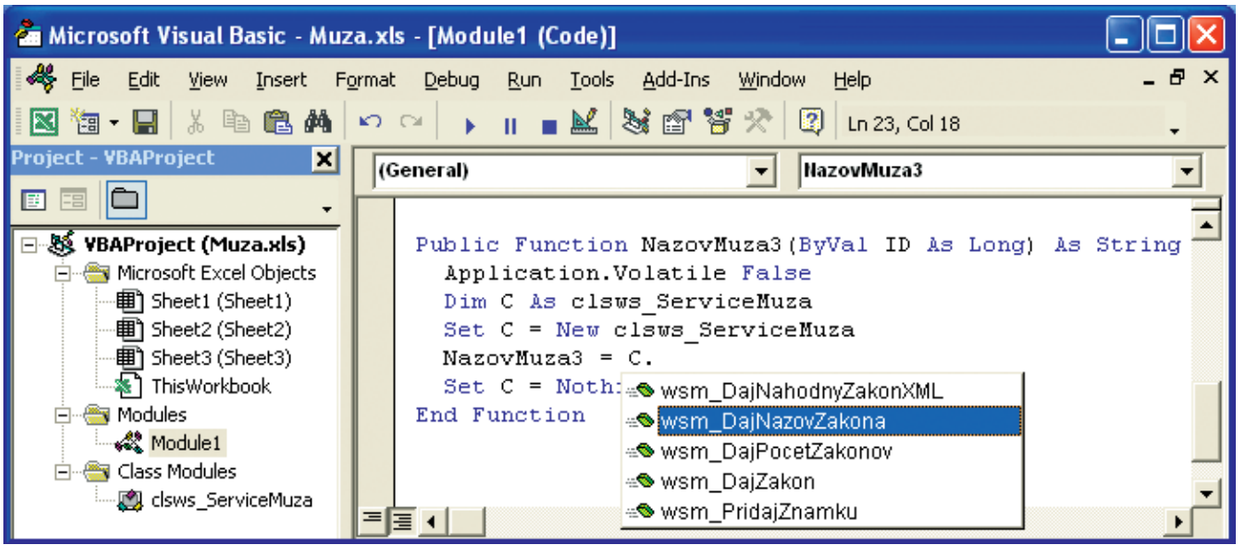
```
'*************************************************************
' This class was created by the Web Service References Tool.
'
' Created: 9/20/2002 03:37:15 PM
'
' Description:
' This class is a Visual Basic for Applications class representation of the Web service
' ServiceMuza
' as defined by http://localhost/WSMuzaXML/ServiceMuza.asmx?WSDL,
'
' This class only contains methods that use simple data types,
' as defined in the WSDL.
'
' To Use:
' Dimension a variable as new clsws_ServiceMuza, and then write code to
' use the methods provided by the class.
```

```
'
' Example:
'    Dim ExampVar as New clsws_ServiceMuza
'    Debug.Print ExampVar.wsm_DajPocetZakonov("Sample Input")
'
' Changes to the code in this class may result in incorrect behavior.
'
'**************************************************************

' Dimensioning private class variables.
Private sc_ServiceMuza As SoapClient
Private Const c_WSDL_URL As String = "http://localhost/WSMuzaXML/ServiceMuza.asmx?WSDL"


Private Sub Class_Initialize()
'**************************************************************
' Subroutine will be called each time the class is instantiated.
' Creates sc_ServiceMuza as new SoapClient, and then
' initializes sc_ServiceMuza.mssoapinit with WSDL file found in
' http://localhost/WSMuzaXML/ServiceMuza.asmx?WSDL.
'**************************************************************

    Set sc_ServiceMuza = New SoapClient
    sc_ServiceMuza.mssoapinit c_WSDL_URL

End Sub

Private Sub Class_Terminate()
'**************************************************************
' Subroutine will be called each time the class is destructed.
' Sets sc_ServiceMuza to Nothing.
'**************************************************************

    'Error Trap
    On Error GoTo Class_TerminateTrap

    Set sc_ServiceMuza = Nothing

Exit Sub

Class_TerminateTrap:
    ServiceMuzaErrorHandler "Class_terminate"
End Sub
Private Sub ServiceMuzaErrorHandler(str_Function As String)
'**************************************************************
' This subroutine is the class error handler. It can be called from any class subroutine
or function
' when that subroutine or function encounters an error. Then it will raise the error along
with the
' name of the calling subroutine or function
'
'**************************************************************

    ' SOAP Error
    If sc_ServiceMuza.faultcode <> "" Then
        Err.Raise vbObjectError, str_Function, sc_ServiceMuza.faultstring
    ' Non SOAP Error
    Else
```

```
 75          Err.Raise Err.Number, str_Function, Err.Description
 76      End If
 77
 78  End Sub
 79
 80  Public Function wsm_DajPocetZakonov() As Long
 81  '************************************************************
 82  ' Proxy function created from http://localhost/WSMuzaXML/ServiceMuza.asmx?WSDL
 83  '************************************************************
 84
 85      'Set error trap
 86      On Error GoTo wsm_DajPocetZakonovTrap
 87
 88      wsm_DajPocetZakonov = sc_ServiceMuza.DajPocetZakonov()
 89
 90  Exit Function
 91  wsm_DajPocetZakonovTrap:
 92      ServiceMuzaErrorHandler "wsm_DajPocetZakonov"
 93  End Function
 94
 95  Public Function wsm_DajZakon(ByVal lng_ID As Long) As String
 96  '************************************************************
 97  ' Proxy function created from http://localhost/WSMuzaXML/ServiceMuza.asmx?WSDL
 98  '************************************************************
 99
110      'Set error trap
101      On Error GoTo wsm_DajZakonTrap
102
103      wsm_DajZakon = sc_ServiceMuza.DajZakon(lng_ID)
104
105  Exit Function
106  wsm_DajZakonTrap:
107      ServiceMuzaErrorHandler "wsm_DajZakon"
108  End Function
109
110  Public Function wsm_DajNahodnyZakonXML() As String
111  '************************************************************
112  ' Proxy function created from http://localhost/WSMuzaXML/ServiceMuza.asmx?WSDL
113  '************************************************************
114
115      'Set error trap
116      On Error GoTo wsm_DajNahodnyZakonXMLTrap
117
118      wsm_DajNahodnyZakonXML = sc_ServiceMuza.DajNahodnyZakonXML()
119
120  Exit Function
121  wsm_DajNahodnyZakonXMLTrap:
122      ServiceMuzaErrorHandler "wsm_DajNahodnyZakonXML"
123  End Function
124
125  Public Function wsm_DajNazovZakona(ByVal lng_ID As Long) As String
126  '************************************************************
127  ' Proxy function created from http://localhost/WSMuzaXML/ServiceMuza.asmx?WSDL
128  '************************************************************
129
130      'Set error trap
131      On Error GoTo wsm_DajNazovZakonaTrap
132
```

```
133     wsm_DajNazovZakona = sc_ServiceMuza.DajNazovZakona(lng_ID)
134
135 Exit Function
136 wsm_DajNazovZakonaTrap:
137     ServiceMuzaErrorHandler "wsm_DajNazovZakona"
138 End Function
139
140 Public Function wsm_PridajZnamku(ByVal lng_ID As Long, ByVal lng_Znamka As Long)
141 As Long
142 '************************************************************
143 ' Proxy function created from http://localhost/WSMuzaXML/ServiceMuza.asmx?WSDL
144 '************************************************************
145
146     'Set error trap
147     On Error GoTo wsm_PridajZnamkuTrap
148
149     wsm_PridajZnamku = sc_ServiceMuza.PridajZnamku(lng_ID, lng_Znamka)
150
151 Exit Function
152 wsm_PridajZnamkuTrap:
153     ServiceMuzaErrorHandler "wsm_PridajZnamku"
154 End Function
```

Porovnajme kód v riadkoch 27, 28, 29, 39, 40 a 133 - tab 6.3 s kódom v tab. 6.1. Vidieť tam podobnosť riešenia. V kóde, ktorý sme dostali pomocou nástroja Web Service References, sú implementované všetky metódy webovej služby - okrem DajNahodnyZakonObj. Táto metóda vracia objekt typu Zakon (viď kapitolu 5). Je to zložitý typ, pre ktorý (zatiaľ) v SOAP toolkite nie je podpora. V implementácii triedy zastupujúcej webovú službu (proxy trieda) v tab. 6.3 si všimnite aj obsluhu výnimiek. Výhodu, ktorú získavame vygenerovanou triedou, ukážeme pri vytvorení používateľskej funkcie NazovMuza3. Pri písaní kódu môžeme s výhodou použiť kontextovú nápovedu - viď obr. 6.10.



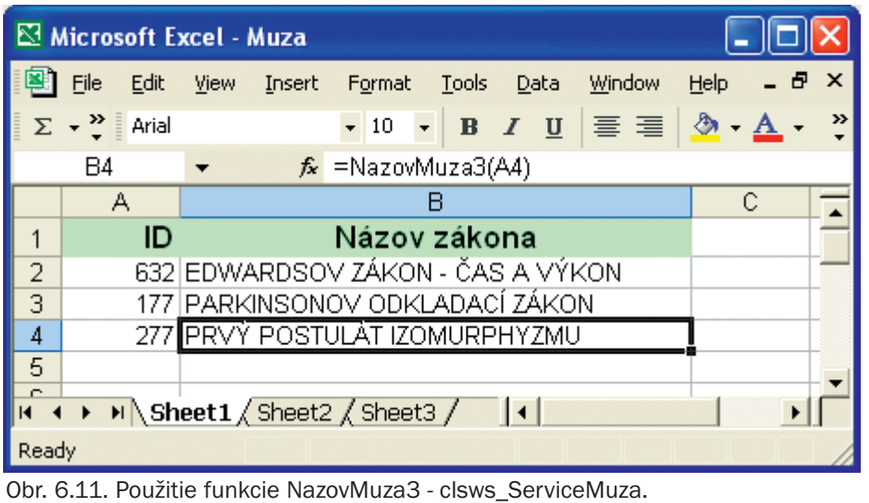
Obr. 6.10. Použitie triedy zastupujúcej webovú službu.

Vytvorená používateľská funkcia NazovMuza3 je v tab. 6.4.

Tab. 6.4. Používateľská funkcia NazovMuza3 pre získanie názvu Murphyho zákona.

```
1  Public Function NazovMuza3(ByVal ID As Long) As String
2    Application.Volatile False
3    Dim C As clsws_ServiceMuza
4    Set C = New clsws_ServiceMuza
5    NazovMuza3 = C.wsm_DajNazovZakona(ID)
6    Set C = Nothing
7  End Function
```

Vo funkcii NazovMuza1 v tab. 6.1 sme vytvorili objekt typu SoapClient. Vo funkcii NazovMuza3 v tab. 6.4 namiesto toho vytvárame objekt proxy triedy, ktorú sme získali pomocou Web Service References. Nemusíme sa starať o inicializáciu SOAP klienta. Stačí volať metódu, ktorá zodpovedá metóde webovej služby. Použitie vytvorenej funkcie NazovMuza3 ukazuje obr. 6.11.



Obr. 6.11. Použitie funkcie NazovMuza3 - clsws_ServiceMuza.

Je zrejmé, že vytvorený dokument nemá veľký praktický význam. V reálnom živote zrejme nebudeme potrebovať v dokumente ukazovať názvy Murphyho zákonov. Ale tak, ako sme v opísanom postupe získali názov Murphyho zákona, môžeme získať z webovej služby stav skladových zásob, resp. cenu výrobku a podobne. Microsoft Excel tak môže byť použitý na strane klienta ako konzument webových služieb.

6.3. Možnosti Microsoft SOAP Toolkit 3.0

V predošlej časti sme spomenuli balík Microsoft SOAP Toolkit 3.0. Ak nepracujete v prostredí Windows XP, po inštalácii SOAP Toolkit budete môcť využiť webové služby v aplikáciách MS Office XP aj 2000, ako bolo opísané v predošlej časti. Možnosti SOAP Toolkit sú však širšie. Umožňuje vytvárať a využívať webové služby v prostredí, ktoré predchádzalo platforme .NET - vo vývojovom prostredí MS Visual Studio 6. S balíkom SOAP Toolkit získate niekoľko zaujímavých príkladov riešených v jazyku Visual Basic 6.0 aj vo Visual C++ 6.0. Ponúka možnosť využiť COM objekty ako webové služby. Slúži k tomu WSDL/WSML Generator. To je na strane servera. Na strane klienta je možnosť využiť knižnicu mssoap30.dll, ako bolo ukázané v predošlej časti. Treba poznamenať, že existujú odporúčania, ktoré nabádajú prejsť od použitia SOAP Toolkitu na platformu .NET (viď napr. článok „Migrating from the SOAP Toolkit to Web Services - Upgrading to Microsoft .NET“ - autor Peter Vogel, dostupný na *http://msdn.microsoft.com/soap*). V prípade, že máte dôvody pre zotrvanie vo využívaní MS Visual Studio 6, alebo potrebujete riešiť „iba“ klientskú časť v aplikáciách MS Office, SOAP Toolkit vám bude užitočným pomocníkom.

7. Budúcnosť XML a webových služieb

Motto: HAWKINSONOV VÝVOJOVÝ ZÁKON
Vývoj neznamená, že nesprávnu teóriu vystrieda správna, ale to, že ju vystrieda taká, ktorej nesprávnosť je menej očividná.

V predošlých kapitolách sme ukázali podstatu XML a jeho úlohu pri tvorbe webových služieb. Dá sa očakávať, že webové služby zohrajú významnú úlohu pri integrácii existujúcich aj pri tvorbe nových aplikácií. Pre ich rozšírenie bude rozhodujúce, ako sa podarí zjednodušiť ich tvorbu pri súčasnom zabezpečení kvality, spoľahlivosti a bezpečnosti. Základom pre to budú stabilné a široko akceptované štandardy.

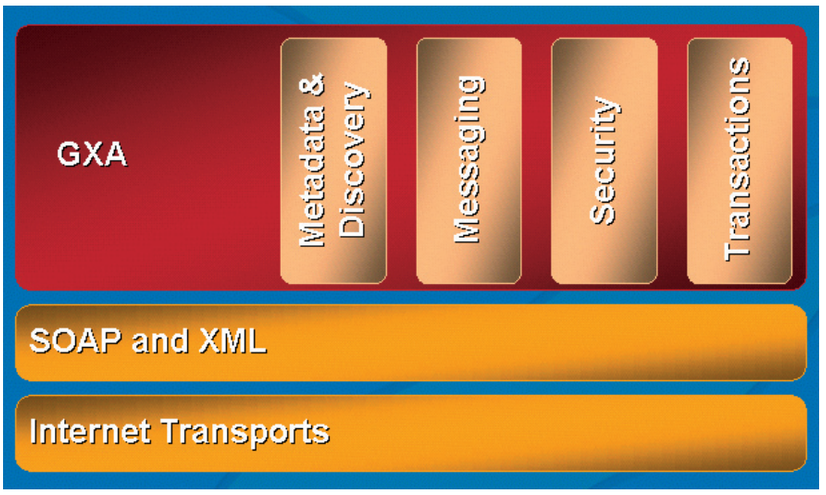
7.1. Základ webových služieb

Od zverejnenia odporúčanie W3C (World Wide Web Consorcium), ktoré definuje XML, uplynulo necelých päť rokov. Bolo publikované vo februári 1998. Odvtedy sme sa dočkali druhej edície odporúčania - v októbri 2000. Stále je však v platnosti verzia 1.0. To svädčí o významnej stabilite základného dokumentu XML. Horšie to bolo s DTD (Document type declaration). Pôvodný návrh je súčasťou odporúčania XML. Od začiatku boli vznášané výhrady voči DTD, najmä preto, že je to jazyk s inými výrazovými prostriedkami, než má XML. Odporúčania, ktoré definujú XML Schema, boli prijaté až v máji roku 2001. Medzitým sa rozšírilo riešenie XDR (XML-Data Reduced), ktoré nemalo základ v odporúčaní W3C. A tak sa v súčasnosti používa DTD, XDR aj XML Schema. K jednoduchosti riešení to neprispieva. Proces prijatia odporúčania W3C je zdĺhavý. Svädčí o tom aj skutočnosť, že Web Services Description Language (WSDL) a Simple Object Access Protocol (SOAP), ktoré tvoria základ webových služieb, ešte stále nedospeli do štádia odporúčania (recommendation) W3C. Sú iba pracovnými skicami (working drafts). Zaujímavý je postup pri tvorbe odporúčania UDDI (Universal Description, Discovery, and Integration). Projekt UDDI začali firmy Microsoft, IBM a Ariba v druhom štvťroku 2000. Verzia 1.0 bola zverejnená v septembri 2000, verzia 2.0 v júni 2001 a v júli 2002 verzia 3.0. Na poslednej verzii sa okrem Microsoftu a IBM podielali aj ďalšie firmy. V copyright tohto dokumentu sú uvedené: Accenture, Ariba, Inc., Commerce One, Inc., Fujitsu Limited, Hewlett-Packard Company, i2 Technologies, Inc., Intel Corporation, International Business Machines Corporation, Microsoft Corporation, Oracle Corporation, SAP AG, Sun Microsystems, Inc., a VeriSign, Inc. Teraz prebiehajú rokovania s W3C a OASIS (Organization for the Advancement of structured Information Standards) o prijatí štandardu UDDI..

7.2. Iniciatíva GXA

Normy XML, SOAP, WSDL nie sú dostatočné pre to, aby bola zaručená bezpečnosť, spoľahlivosť, rozširovateľnosť, správa webových služieb. A tak Microsoft a IBM vyšli s iniciatívou GXA - Global XML Web Services Architecture. Východiskom pre GXA je pevne definovaný základ - transportné prostriedky internetu, XML a SOAP. Základným cieľom je zabezpečiť otvorenosť, t.j. možnosť definovať štandardy, ktoré umožnia vytvárať univerzálne stavebné prvky pre softvérové riešenia a schopnosť zahrnúť nové štandardy, ktorých potrebu si vyžiada reálne nasadzovanie webových služieb. Vyjadruje to obr. 7.1. V GXA je snaha nájsť riešenie spoločných problémov, ktoré súvisia s tvorbou webových služieb, ako je:

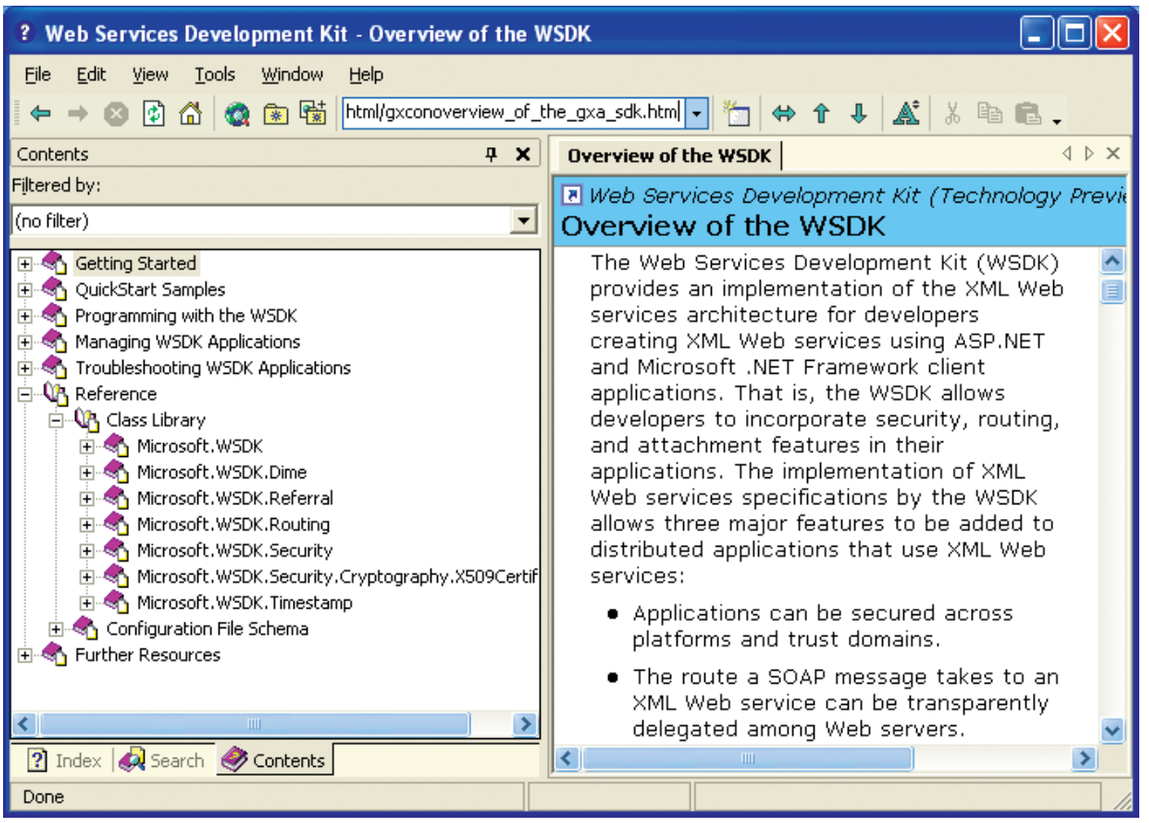
- Spoľahlivé posielanie správ cez nespoľahlivé a zmenám podliehajúce prostredie internetu. Riešia to návrhy noriem WS-Routing a WS-Referral.
- Zabezpečenie integrity, celistvosti, utajenia a autentifikácie obsahu posielaných správ. Rieši to návrh normy WS-Security.
- Špecifikácia XML formátu pre podporu vyhľadávania webových služieb. Rieši to návrh jazyka WS-Inspection.
- Vytvorenie rozšíriteľnej množiny protokolov, ktoré zabezpečia koordináciu akcií distribuovaných aplikácií. Je to obsahom normy WS-Coordination. Dva konkrétne typy koordinácie (Atomic Transaction a Business Activity) špecifikuje návrh normy WS-Transaction.



Obr. 7.1. GXA - Global XML Web Services Architecture.

Niektoré návrhy pochádzajú iba z dielne Microsoftu, napr. WS-Referral, WS-Routing. Niektoré sú spoločným dielom Microsoftu a IBM, napr. WS-Inspection. Na niektorých návrhoch sa podieľajú aj ďalšie firmy, napr. WS-Security je spoločným dielom firiem IBM, Microsoft a VeriSign. WS-Coordination a WSTransaction predložili Bea Systems, IBM a Microsoft.

Základnou zásadou firiem spolupracujúcich na tvorbe normatívnych dokumentov GXA je - spolupráca pri tvorbe štandardov, konkurencia pri ich implementácii. Pre podporu vývoja webových služieb v ASP.NET a .NET Framework Microsoft zverejnil Web Services Development Kit (WSDK). Je dostupný na adrese *http://msdn.microsoft.com/downloads*. Sú v ňom k dispozícii triedy, ktoré predstavujú implementáciu niektorých súčastí GXA - viď obr. 7.2.



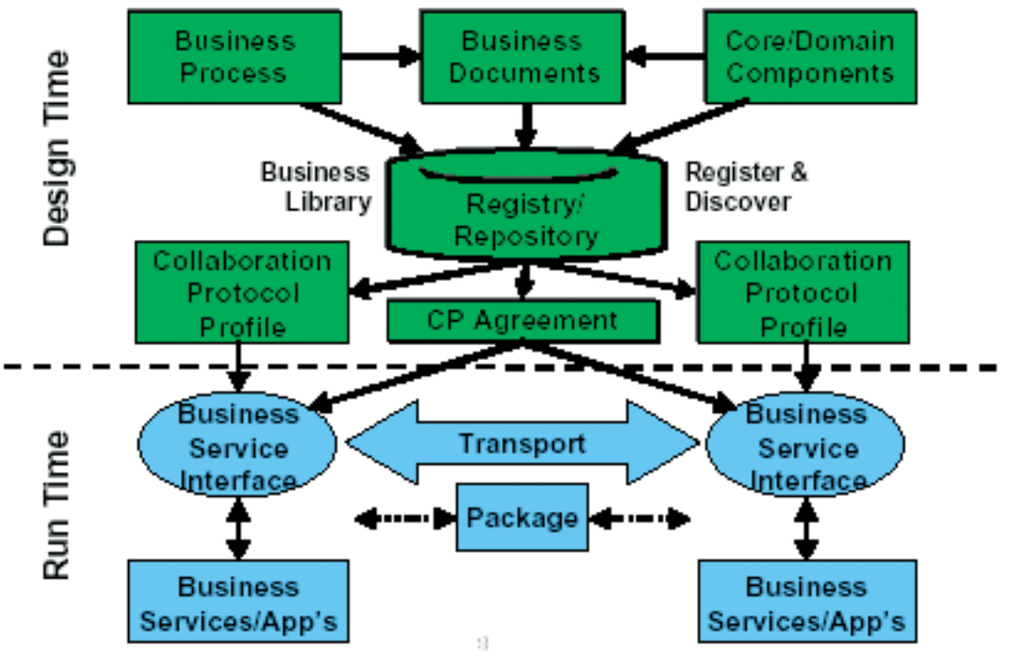
Obr. 7.2. WSDK - Web Services Development Kit.

V podobe WSDK je tak v relatívne krátkej dobe od zverejnenia dokumentov GXA k dispozícii možnosť ich verifikácie v konkrétnej implementácii.

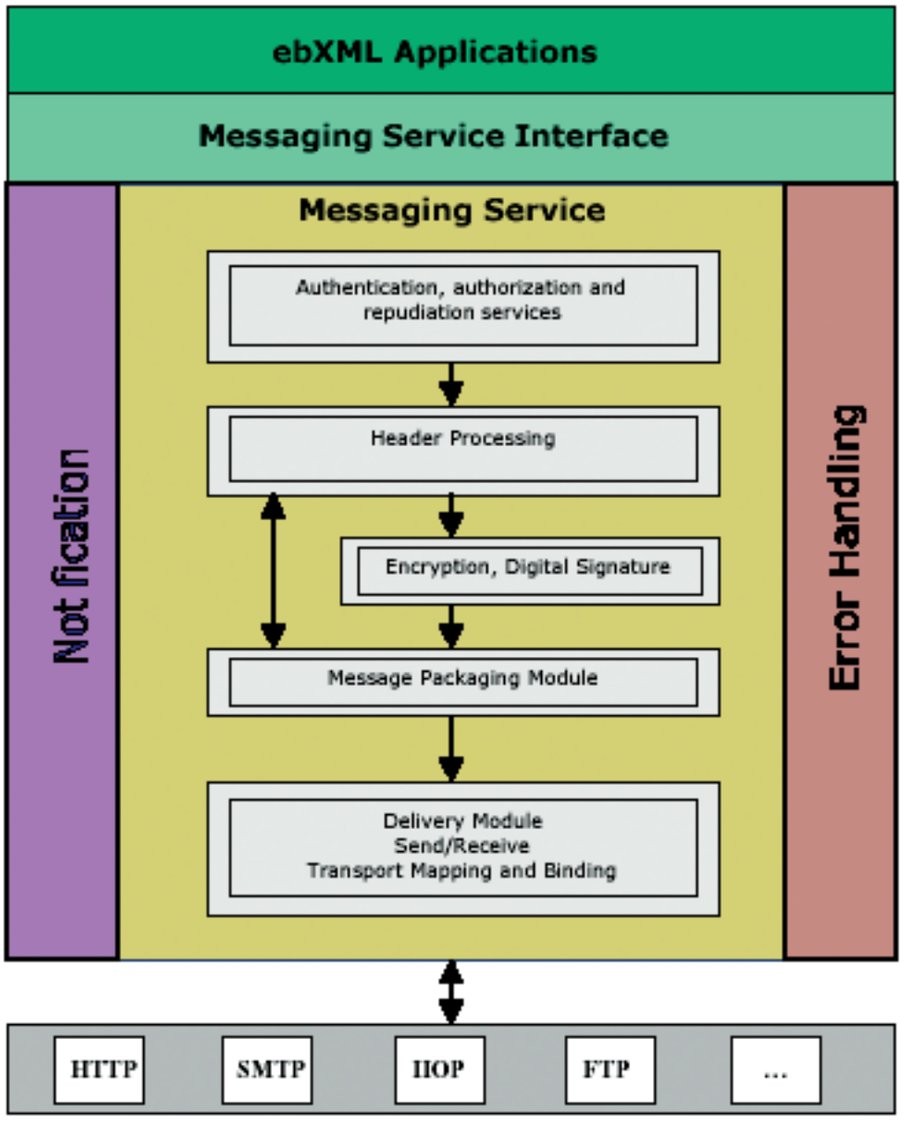
7.3. Iné iniciatívy

GXA nie je jedinou iniciatívou, ktorá má za cieľ vytvoriť štandardy pre podporu webových služieb. Na pôde W3C bola ustanovená pracovná skupina Web Services Architecture (WSA). V apríli 2002 bola uverejnená prvá pracovná skica (Working Draft), opisujúca požiadavky na architektúru webových služieb - Web Services Architecture Requirements. Posledné upresnenie tohto dokumentu bolo uverejnené v auguste 2002. Podobne ako v GXA aj tu sa kladie za cieľ zabezpečiť interoperiabilitu, spoľahlivosť, bezpečnosť, škálovateľnosť a rozšíriteľnosť webových služieb pri súčasnom zabezpečení ich konzistencie s webom. Pretože v tejto pracovnej skupine sú zástupcovia Microsoftu aj IBM, dá sa predpokladať snaha presadiť normy GXA do WSA.

Významné aktivity vo vytváraní normatívnych dokumentov založených na XML sa robia na pôde OASIS (Organization for the Advancement of Structured Information Standards). Výsledkom jeho technického výboru UBL (Universal Business Language) je súbor špecifikácií súhrnne označovaných ako ebXML (Electronic Business XML). Pozrime sa na dva obrázky vybraté z týchto dokumentov (obr. 7.3 a 7.4). Vidieť podobnú terminológiu, ako je použitá v GXA, aj náčrt riešenia podobných problémov.



Obr. 7.3. Referenčný model ebXML pre etapu návrhu a behu aplikácií.



Obr. 7.4. Architektúra obsluhy správ v ebXML.

Očakáva sa, že dôležitú úlohu pri rozširovaní technológie webových služieb zohrá organizácia **WS-I - Web Services Interoperability.** Začiatkom februára 2002 ju založili firmy Microsoft, IBM, BEA Systems a Intel. Dnes sú jej členmi takmer všetky rozhodujúce organizácie a firmy, ktoré sa podieľajú na rozvoji, nasadení a použití webových služieb. Spomeňme aspoň AT&T, Borland, ESRI, Oracle, Rational Software, SAP, Softronic, Sybase, VeriSign, Unisys, ale aj United Airlines.

Organizácia WS-I má v súčasnosti postavené tri hlavné ciele:

- poskytovať poradenstvo a vzdelávanie pri implementácii a nasadení webových služieb,
- podporovať interoperabilitu webových služieb, ich nezávislosť od platforiem, operačných systémov a programovacích jazykov,
- podporovať spoločné vízie interoperability webových služieb, aby zákazníci mali uľahčené rozhodovanie, aby rástlo uplatnenie webových služieb a pokračoval vývoj technológií webových služieb.

Príspevkom k dosiahnutiu týchto cieľov je tvorba tzv. profilov - pomenovaných kolekcií špecifikácií a podmienok, pri ktorých ich možno použiť. Aj v tejto brožúre sme poukazovavali na postup tvorby narmatívnych dokumentov, ktoré sa pri webových službách používajú. Nielenže tieto normy sú z dielní rôznych normotvorných organizácií, ale aj v jednej organizácii sú už dnes k dispozícii rôzne verzie. A to je normotvorný proces iba v začiatkoch. Je preto veľmi dôležité vedieť, ktoré normatívne dokumenty je možné vzájomne kombinovať. Ako príklad uveďme profil WS-I Basic Web services:

- XML Schema 1.0
- SOAP 1.1
- WSDL 1.1
- UDDI 2.0

Okrem profilov budú na pôde WS-I pripravované testovacie postupy a testovacie nástroje, budú tvorené príklady aplikácií a poskytované ich zdrojové kódy. Dá sa teda očakávať, že materiály, ktoré bude organizácia WS-I uverejňovať, budú užitočné pre vývojárov, systémových integrátorov, poskytovateľov, ale aj pre používateľov webových služieb. Už dnes sa dajú získať zaujímavé informácie na adrese *http://www.ws-i.org*.

Ťažko predvídať, ktoré zo spomenutých iniciatív sa v konečnom dôsledku presadia. Jedno je však isté. XML a technológie, ktoré sú na ňom založené, ovládnu web. Ponúknu možnosť vyriešiť problémy vzájomnej spolupráce sýstémov rôznych platforiem. Budú prostriedkom pre automatizáciu nielen obchodných činností, ale aj spracovania neustále rastúceho objemu údajov, ktorými ľusdtvo disponuje.

Dodatok 1

Definícia niektorých pojmov

Motto: ZÁKON PÁNA COOPERA
Ak nerozumieš niektorému odbornému výrazu, nemaj strach!
Článok zostane aj bez neho úplne zrozumiteľný.

SGML

Standard Generalized Markup Language - štandard ISO (International Organization for Standardization) s číselným označením 8879 z roku 1986 pre výmenu údajov medzi rôznymi počítačmi.

W3C

World Wide Web Consorcium - medzinárodné združenie organizácií pre tvorbu a správu štandardov pre World Wide Web. Zverejňuje ich na adrese *http://www.w3.org/TR*.

HTML

HyperText Markup Language - pôvodný návrh Tim Berners-Lee, potom odporúčanie W3C, posledná verzia 4.0 uverejnená v apríli 1998, aplikácia SGML pre zverejňovanie dokumentov na webe.

XML

Extensible Markup Language - odporúčanie W3C uverejnené vo februári 1998, druhá edícia uverejnená v októbri 2000, zjednodušenie SGML.

DTD

Document Type Declaration - spôsob deklarovania typov dokumentu obsiahnutý v XML, jazyk s inými výrazovými prostriedkami, než má XML.

XML-Data

Záznam (Note) W3C z januára 1998, ktorého cieľom bolo definovať spôsob deklarácie typov dokumentu XML výrazovými prostriedkami XML.

XDR

XML-Data Reduced - definovanie typov dokumentu vychádzajúce z XML-Data používaný vo viacerých aplikáciách.

XML Schema

Odporúčanie W3C uverejnené v máji 2001. Umožňuje definovať typy dokumentu XML výrazovými prostriedkami XML. Pozostáva z troch častí: Part 0 Primer, Part 1 Structures, Part 2 Datatypes.

Namspaces in XML

Odporúčanie W3C uverejnené v januári 1999. Definuje spôsob pomenovania elementov XML dokumentu tak, aby nevznikali konflikty - nejednoznačnosti.

Web Service - webová služba

Softvérová aplikácia identifikovaná prostredníctvom URI (Uniform Resource Identifier), ktorej interfejsy a väzby je možné definovať, opísať a vyhľadávať ako artefakty XML. Podporuje priamu interakciu s inými softvérovými aplikáciami prostredníctvom správ zapísanými v jazyku XML a prenášanými protokolmi internetu.

WSDL

Web Services Description Language - jazyk na opis webových služieb, založený na XML. Odporúčanie pripravované W3C, v júli 2002 pracovná skica (working draft in development).

SOAP

Simple Object Access Protocol - protokol založený na XML, určený pre výmenu údajov v decentralizovanom distribuovanom prostredí. Pripravované odporúčanie W3C, v júni 2002 v štádiu pracovných skicí (working drafts in last call) SOAP Version 1.2 Part 0: Primer, Part 1: Messaging Frame, Part 2: Adjuncts.

UDDI

Universal Description, Discovery, and Integration. Projekt firiem Microsoft, IBM a Ariba začatý v druhom štvťroku 2000. Verzia 1. v septembri 2000, verzia 2.0 v júni 2001, v júli 2002 verzia 3.0. Webová služba pre opis, vyhľadávanie a integráciu webových služieb.

GXA

Global XML Web Services Architecture - iniciatíva firiem Microsoft a IBM na poli tvorby špecifikácií pre webové služby. Predložené návrhy: WS-Routing, WS-Referral, WS-Security, WS-Inspection, WS-Coordination, WS-Transaction.

Vydal: Microsoft, s.r.o., Novodvorská 1010/14B, 140 00 Praha 4, tel.: +420 261 197 111, fax: +420 260 197 100
http://www.microsoft.com/cze, http://www.microsoft.com/slovakia