Anhang 4

Arbeiten mit Webdiensten in VFP 7.0

Die nächste Generation von Webanwendungen erwartet den Gebrauch von Webdiensten. Visual FoxPros neue XML-Fähigkeiten und Webserver-Erweiterungen vereinfachen die Erstellung von Webdiensten, die von jedem Punkt im Internet aufgerufen werden können.

Bevor wir uns mit den Webdiensten selbst befassen, sollten Sie die Grundlagen von XML verstehen, wie ist in VFP 7 integriert ist und wo Sie es in Ihren Anwendungen einsetzen können.

Einführung in XML

Kurz gesagt ist XML ein (relativ) neues Datenformat. Warum also die ganze Aufregung? In erster Linie ist XML von allen Anbietern der wichtigsten Plattformen akzeptiert und bietet damit die Möglichkeit für den Austausch von Daten über die Grenzen der Plattformen hinweg.

XML weist eine große Ähnlichkeit mit HTML auf, und das aus gutem Grund – beide sind aus der gleichen Definitionssprache entwickelt – der Standard Generalized Markup Language (SGML). XML enthält Tags (die Worte in spitzen Klammern), die Textblöcke abschließen. Anders als HTML, wo jeder Tag eine spezielle Bedeutung hat, ist XML erweiterbar – die Tags haben keine festgelegte Bedeutung. Sie können den Tags Ihre eigene Definition zuordnen.

Zum Beispiel hat bei HTML das Tag <TABLE> eine besondere Bedeutung. Darauf aufbauend kann jeder Browser, der die HTML-Spezifikation versteht, Tabellen korrekt darstellen. Im Gegensatz dazu kann das Tag <TABLE> in XML jede Bedeutung haben. Zum Beispiel steht das Tag <TABLE> für ein Möbelstück – einen Esstisch, einen Cafétisch oder einen Kartentisch:

```
<FURNITURE>
      <TABLE>"Dining Room"</TABLE>
      <TABLE>"Coffee"</TABLE>
      <TABLE>"Card"</TABLE>
</FURNITURE>
```

Eine detaillierte Beschreibung von XML finden Sie in MSDN unter dem Eintrag „XML Tutorial“.

Die Struktur eines XML-Dokuments

Hier ein Beispiel eines XML-Dokuments:

```
<?xml version = "1.0" encoding="Windows+1252" standalone="yes"?>
<VFPData>
      <customer>
         <iid>1</iid>
         <cacctno>001000</cacctno>
         <cname>Journey Communications</cname>
         <caddress1>101 Main St.</caddress1>
         <ccity>Richmond</ccity>
         <cstate region="1">VA</cstate>
         <czip>22901</czip>
         <ctype/>
      </customer>
      <customer>
         <iid>4</iid>
         <cacctno>001003</cacctno>
         <cname>Sergio Vargas, Attorney at Law</cname>
         <caddress1>115 Pacific Coast Highway</caddress1>
         <ccity>Malibu</ccity>
         <cstate region="1">CA</cstate>
         <czip>80766</czip>
         <ctype/>
      </customer>
</VFPData>
```

Die erste Zeile des Dokuments ist die XML-Deklaration:

```
<?xml version = "1.0" encoding="Windows+1252" standalone="yes"?>
```

Diese Deklaration ist nicht erforderlich, aber es ist empfehlenswert, sie an den Anfang aller XML-Dokumente zu stellen. Sie identifiziert das Dokument als XML-Dokument und gibt die XML-Version an, mit der es erstellt wurde. Die Anweisung „standalone“ gibt an, ob das XML-Dokument alleine steht oder ob es zusätzliche Definitionen aus einer anderen Datei benötigt. Beachten Sie, dass das Wort „xml“ klein geschrieben sein muss. XML-Tags unterscheiden im Gegensatz zu HTML zwischen Groß- und Kleinschreibung.

Die nächste Zeile des Dokuments ist der Anfangstag des Root-Elements:

```
<VFPData>
```

Die XML-Spezifikation erfordert, dass alle XML-Dokument ein einzelnes Root-Element enthält, das alle anderen Elemente des Dokuments einschließt.

Innerhalb des Root-Elements befinden sich zwei Elemente vom Typ <customer>, die jeweils einen individuellen Kundendatensatz repräsentieren. Innerhalb dieser Elemente befinden sich neun andere Elemente, die jeweils ein Feld innerhalb des Datensatzes repräsentieren.

Elemente

Elemente sind die gebräuchlichste Markierungsform, die in einem XML Dokument auftritt. Sie bestehen aus einem öffnenden und einem schließenden Tag sowie Inhalten zwischen den Tags (auch Daten genannt) sowie optionalen Attributen mit dazugehörenden Werten (vgl. dazu den nächsten Abschnitt). Ein Beispiel:

```
<cname>Sergio Vargas, Attorney at Law</cname>
```

Wenn ein bestimmtes Element keinen Inhalt hat, können Sie ein einzelnes Tagelement verwenden um es zu repräsentieren. Der Tag ctype beispielsweise enthält keine Daten und wird folgendermaßen repräsentiert:

```
<ctype/>
```

Auf diese Weise wird sowohl der öffnende als auch der schließende Tag in einem einzigen Element vereint.

Attribute

Ein Attribut ist ein Paar zusammengesetzt aus dem Namen und dem dazugehörenden Wert, die innerhalb des öffnenden Tag zusammengeschlossen sind. Ein Beispiel:

```
<cstate region="1">CA</cstate>
```

In diesem Element bildet region=“5“ das Paar aus Namen und Wert, das das Attribut bildet. Attribute können eingesetzt werden, um zusätzliche Informationen über ein bestimmtes Element zu bieten.

Kommentare

In einem XML-Dokument haben Kommentare das folgende Format:

```
<!—This is a comment -->
```

Kommentare werden von Parsern ignoriert.

Gut formatiertes XML

Wenn sie sich an die Grundregeln von XML halten, sind XML-Dokumente gut formatiert. So schreibt die XML-Spezifikation beispielsweise vor, dass XML-Dokumente einen eindeutigen Root-Knoten enthalten müssen. Eine

andere Regel des XML besagt, dass Anfangs- und Endtag identisch sein sollten – einschließlich der Groß- und Kleinschreibung. Als Beispiel sind dies gültige XML-Tags:

```
<cacctno>001003</cacctno>
```

Die folgenden Tags entsprechen der Norm nicht, da sich Groß- und Kleinschreibung unterscheiden:

```
<cacctno>001000</cAcctNo>
```

Mit HTML können sie sehr lax mit den schließenden Tags umgehen. Browser, die mit HTML umgehen, können gut vergeben – fehlt ein schließender Tag kann der Browser dies kompensieren und das HTML trotzdem in eine ansehnliche Form bringen. XML-Parser sind da unnachgiebiger. Für jeden öffnenden Tag muss auch ein schließender vorhanden sein.

Reservierte Zeichen

Es gibt in XML einige reservierte Zeichen, die innerhalb des Datenteils nicht so genutzt werden können wie sie sind, da sie nach der XML-Spezifikation eine besondere Bedeutung haben. Nehmen wir als Beispiel die öffnende spitze Klammer, die in XML als Markierung für den Anfang eines Tags genutzt wird. Wenn Sie innerhalb Ihrer Daten dieses Zeichen haben, müssen Sie es ersetzen: <.

In Tabelle 1 finden Sie eine Aufstellung der reservierten Zeichen und deren Sequenz für die Ersetzung.

Tabelle 1. Diese Tabelle zeigt die häufig gebrauchten reservierten Zeichen, die durch die entsprechenden Sequenzen ersetzt werden müssen.

Reserviertes Zeichen	Sequenz für die Ersetzung
&	&amp;
<	<
>	>
“	"
’	'

Ein Beispiel: in XML muss der Text „Liederbach & Associates“ muss folgendermaßen dargestellt werden:

```
<cname>Liederbach &amp; Associates</name>
```

Elementzentriertes und attributzentriertes XML

Visual FoxPro 7 kann XML-Daten in drei unterschiedlichen Formaten erzeugen:

- Elementzentriert
- Attributzentriert
- Rohes XML

Die folgenden Abschnitte zeigen, wie eine Kundentabelle mit der in Tabelle 2 gezeigten Struktur in jedem dieser drei Formate repräsentiert wird.

Tabelle 2. Diese Tabelle zeigt die Struktur einer Kundentabelle, die in den XML-Beispielen dieses Kapitels benutzt wird.

Name	Typ	Größe
iID	Integer	4
cAcctNo	Character	6
cName	Character	50
cAddress1	Character	40
cCity	Character	25
cState	Character	2
cZip	Character	10

Elementzentriertes XML

Mit elementzentriertem XML formatiert sehen die Datensätze folgendermaßen aus:

```
<?xml version = "1.0" encoding="Windows-1252" standalone="yes"?>
<VFPData>
      <customer>
         <iid>1</iid>
         <cacctno>001000</cacctno>
         <cname>Journey Communications</cname>
         <caddress1>101 Main St.</caddress1>
         <ccity>Richmond</ccity>
         <cstate>VA</cstate>
         <czip>22901</czip>
      </customer>
      <customer>
         <iid>4</iid>
         <cacctno>001003</cacctno>
         <cname>Sergio Vargas, Attorney at Law</cname>
         <caddress1>115 Pacific Coast Hwy</caddress1>
```

```
        <ccity>Malibu</ccity>
        <cstate>CA</cstate>
        <czip>80766</czip>
      </customer>
</VFPData>
```

In diesem Format wird jeder Datensatz durch ein individuelles Element repräsentiert. Jedes Feld wird als ein Element dargestellt, dass in das Datensatzelement eingebettet ist.

Attributzentriertes XML

Mit elementzentriertem XML formatiert sehen die Datensätze folgendermaßen aus:

```
<?xml version = "1.0" encoding="Windows-1252" standalone="yes"?>
<VFPData>
      <customer iid="1" cacctno="001000" cname="Journey
Communications" caddress1="101 Main St." ccity="Richmond"
cstate="VA" czip="22901"/>
      <customer iid="4" cacctno="001003" cname="Sergio Vargas,
Attorney at Law" caddress1="115 Pacific Hwy" ccity="Malibu"
cstate="CA" czip="80766"/>
</VFPData>
```

In diesem Format wird jeder Datensatz durch ein Element mit nur einem Tag repräsentiert (jedes Element erhält den gleichen Namen wie der dazugehörende Cursor.) Alle Feldelemente werden als Attribut eines Datensatzelements dargestellt. Alle Feldwerte werden als Strings dargestellt, unabhängig von ihren Original-Datentyp. Wie Sie sehen können, ist das attributzentrierte Format kürzer als das elementzentrierte.

Rohes XML

Mit rohem XML formatiert sehen die Datensätze folgendermaßen aus:

```
<?xml version = "1.0" encoding="Windows-1252" standalone="yes"?>
<VFPData>
      <row iid="1" cacctno="001000" cname="Journey Communications"
caddress1="101 Main St." ccity="Richmond" cstate="VA" czip="22901"/>
      <row iid="4" cacctno="001003" cname="Sergio Vargas, Attorney at
Law" caddress1="115 Pacific Hwy" ccity="Malibu" cstate="CA"
czip="80766"/>
</VFPData>
```

Dieses Format ist mit dem attributzentrierten identisch, lediglich wird jedes Datensatzelement mit „row“ benannt.

Schemata und gültiges XML

Wie im Abschnitt „Gut formatiertes XML“ beschrieben, muss ein XML-Dokument korrekt formatiert sein, damit es von einem Parser korrekt ausgewertet werden kann. Zusätzlich zur korrekten Formatierung kann ein XML-Dokument gültig sein, wenn es einem angegebenen Schema zugeordnet ist.

Ein Schema definiert die Struktur und Bedeutung eines XML-Dokuments einschließlich der Elementnamen, Typen und Attribute. Sie können ein Schema einsetzen, um das Format festzulegen, dass Sie für ein XML-Dokument erwarten.

Schemata können in einer separaten Datei gespeichert oder in ein XML-Dokument eingeschlossen werden. Hier ein Beispiel für ein eingebettetes Schema:

```
<?xml version = "1.0" encoding="Windows-1252" standalone="yes"?>
<VFPData>
     <xsd:schema id="VFPSchema"
xmls:xsd=http://w3.org/2000/10/XMLSchema xmlns:msdata="urn:schemas-
microsoft-com:xml-msdata">
        <xsd:element name="customer" minOccurs="1"
maxOccurs="unbounded">
           <xsd:complexType>
              <xsd:attribute name="iid" use="required"
type="xsd:int"/>
              <xsd:attribute name="cacctno" use="required">
                 <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                       <xsd:maxLength value="6"/>
                    </xsd:restriction>
                 </xsd:simpleType>
              </xsd:attribute>
              <xsd:attribute name="cname" use="required">
                 <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                       <xsd:maxLength value="50"/>
                    </xsd:restriction>
                 </xsd:simpleType>
              </xsd:attribute>
              <xsd:attribute name="caddress1" use=required">
                 <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                       <xsd:maxLength value="40"/>
                    </xsd:restriction>
                 </xsd:simpleType>
              </xsd:attribute>
              <xsd:attribute name="ccity" use=required">
                 <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                       <xsd:maxLength value="25"/>
                    </xsd:restriction>
                 </xsd:simpleType>
              </xsd:attribute>
```

```
            <xsd:attribute name="cstate" use=required">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                     <xsd:maxLength value="2"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="czip" use=required">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                     <xsd:maxLength value="10"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:attribute>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="VFPData" msdata:lsDataSet="true">
         <xsd:complexType>
            <xsd:coice maxOccurs="unbounded">
               <xsd:element ref="customer"/>
            </xsd:choice>
         </xsd:complexType>
      </xsd:element>
   </xsd:schema>
   <customer iid="1" cacctno="001000" cname"Journey
Communications" caddress1="101 Main St." ccity="Richmond"
cstate="VA" czip="22901"/>
   <customer cacctno="001003" cname"Sergio Vargas, Attorney of
Law" caddress1="115 Pacific Coast Hwy" ccity="Malibu" cstate="A"
czip="80766"/>
</VFPData>
```

Wenn Sie sich das Inline-Schema etwas genauer ansehen, werden Sie die angegebenen Elementnamen wieder finden, ob sie erforderlich sind sowie deren Typ. Visual FoxPro 7 unterstützt den W3C XSD-Schemaformat. Details finden Sie unter http://www.w3.org/TR/2000/CR-xmlschema-0-2001024/.

Eine Tabelle mit den Informationen, wie die Datentypen von Visual FoxPro in XSD-Typen abgebildet werden, lesen Sie in der Hilfedatei von VFP 7 den Eintrag „XML-Funktionen“.

XML Namespaces

Namespaces sind eine Möglichkeit, Elemente in einem XML-Dokument eindeutig zu identifizieren. Dies ist wichtig, da der Autor eines XML-Dokuments dessen XML-Tags definiert und wenn Sie Dokumente mehrerer Autoren mischen, laufen Sie in Gefahr einer Kollision der Namen der Tags.

Sie können einen oder mehrere Namespaces für ein Dokument definieren, indem Sie das Attribut xmlns in einem öffnenden SML-Elementtag benutzen. Das Namespace gilt für den gesamten Inhalt eines Elements. Da der Namespace für die eindeutige Identifizierung der Tags genutzt wird, muss der

Namespace selbst auch eindeutig sein. Häufig setzen die Autoren von XML-Dokumenten die URL ihres Unternehmens als Namespace ein, da diese garantiert eindeutig ist. Ein Beispiel:

```
<VFPData xmls="www.oakleaf.com">
     <customer iid="1" cacctno="001000" cname="Journey
Communications" caddress1="101 Main St." ccity="Richmond"
cstate="VA" czip="22901"/>
     <customer iid="4" cacctno="001003" cname="Sergio Vargas,
Attorney at Law" caddress1="115 Pacific Hwy" ccity="Malibu"
cstate="CA" czip="80766"/>
</VFPData>
```

XML-Funktionen in Visual FoxPro 7

Obwohl Sie Ihre eigenen Funktionen schreiben könnten, die einen Cursor in XML und zurück umwandeln, kannte VFP 6 XML nicht. Die hat sich in Visual FoxPro 7 durch die Einführung von drei neuen Funktionen geändert: CursoToXML(), XMLToCursor() und XMLUpdateGram().

CursorToXML()

Die neue Funktion CursorToXML() tut genau das, was ihr Name sagt – sie konvertiert einen Cursor von Visual FoxPro in XML. Dieser Befehl kann in Geschäftsobjekten eingesetzt werden, um interne Cursor in XML umzusetzen, das vom Client verarbeitet werden kann.

Die Syntax von CursorToXML() ist:

```
CursorToXML(nWorkArea | cTableAlias, cOutput [, nOutputFormat [,
nFlags [, nRecords [, cSchemaName [, cSchemaLocation [, cNameSpace
]]]]]])
```

Im ersten Parameter können Sie wahlweise den Arbeitsbereich oder den Alias des Cursors angeben, der umgewandelt werden soll. Der Parameter cOutput gibt den Namen einer Speichervariablen oder Datei an, in der das Ergebnis in XML gespeichert wird. Um einen Dateinamen anzugeben, müssen Sie den Parameter nFlags entsprechend setzen (Details finden Sie in der Hilfedatei von VFP 7 unter dem Eintrag „CURSORTOXML()-Funktion).

Der Parameter nOutputFormat gibt das XML-Ausgabeformat an (1 – ELEMENTS, 2 – ATTRIBUTES oder 3 – RAW).

Mit den Parametern cShemaName, cSchemaLocation und cNameSpace legen Sie Angaben über Schemata und Namespaces fest. Der Parameter nFlags ermöglicht Ihnen das Festlegen weiterer Ausgabeoptionen wie das Einfügen von leerem Raum, Kodierung der Ausgabe sowie die Formatierung leerer Elemente.

Die folgende Anweisung wandelt den Inhalt der Tabelle Customer in elementzentriertes XML um und speichert das Ergebnis in der Speichervariablen lcXML:

```
CURSORTOXML("Customer", "lcXML")
```

Die nächste Anweisung wandelt den Inhalt der Tabelle Customer in attributzentriertes XML um und speichert das Ergebnis in Datei Results.XML und generiert ein Inlineschema mit dem Namespace www.microsoft.com:

```
CURSORTOXML("Customer", "Results.xml", 2, 512, 0,"1", "",
"www.microsoft.com")
```

XMLToCursor

Die neue Funktion XMLToCursor() wandelt einen XML-String in einen Cursor von Visual FoxPro um. Dieser Befehl kann in Geschäftsobjekten eingesetzt werden, um eine von einem Client empfangene XML-Eingabe in VFPs internen Cursor umzusetzen.

Die Syntax von XMLToCursor() ist:

```
XMLTOCURSOR(XMLSource eExpression | cXMLFile [, cCursorName [,
nFlags ]])
```

Der Parameter cXMLExpression gibt einen XML-Text oder einen Ausdruck an, der als gültige XML-Daten bewertet wird. Dabei kann es sich um eines der folgenden handeln:

- Eine Speichervariable.
- Ein Memofeld.
- Die Rückgabe einer HTTP-Anfrage.
- Die Rückgabe eines SOAP-Methodenaufrufs (vgl. dazu den Abschnitt „Simple Object Access Protokoll (SOAP) weiter hinten in diesem Kapitel).
- XML von XML DOM (Document Object Model). Das XML DOM ist eine Programmierschnittstelle für XML-Dokumente, die Ihnen den Zugriff auf ein XML-Dokument in einer Baumansicht ermöglicht.
- Ein ADO-Strem.

Alternativ dazu können Sie im Parameter cXMLFile den Namen einer XML-Datei angeben. Der Parameter nFlags gibt an, wie der erste Parameter behan-

delt wird. Einzelheiten erfahren Sie in der Tabelle in der Hilfedatei von VFP 7 unter dem Eintrag „XMLToCursor()“.

Der Parameter cCursorName enthält den Namen des Cursors, in dem das Ergebnis gespeichert werden soll. Wird dieser Cursor bereits benutzt, generiert VFP einen Fehler. XMLToCursor() kann XML mit und ohne Schema umwandeln.

Die folgende Anweisung wandel die Datei Results.xml in den Cursor „MyCursor“ um:

```
XMLTOCURSOR("Results.xml", "MyCursor", 512)
```

XMLUpdateGram()

Die neue Funktion XMLUpdateGram() liefert einen XML-String zurück, der alle Änderungen in einer gepufferten Tabelle oder Cursor enthält. Sie müssen die Tabellenpufferung und SET MULTILOCKS ON einstellen, bevor Sie XMLUpdateGram() einsetzen können.

Die Syntax von XMLUpdateGram() ist:

```
XMLUPDATEGRAM( [cAliasList [, nFlags]])
```

Der Parameter cAliasList gibt eine kommaseparierte Liste offener Tabellen oder Cursor an, die das Updategram enthalten soll. Ist cAliasList leer schließt XMLUpdateGram() alle geöffneten Tabellen und Cursor der aktuellen Datensitzung ein.

Der Parameter nFlags enthält eine Liste kumulierender Flags, mit denen die Formatierung der XML-Ausgabe festgelegt wird. In der Hilfedatei von VFP 7 finden Sie unter dem Eintrag „XMLUPDATEGRAM()-Funktion“ eine Tabelle, in der diese Flags beschrieben werden.

Ändern von Datensätzen

Um die Arbeitsweise von XMLUpdateGram() beim Ändern von Datensätzen zu demonstrieren, rufen Sie Visual FoxPro auf, wechseln zum Verzeichnis ProjectX\Data dieses Kapitels und geben im Befehlsfenster das Folgende ein:

```
USE Customer
SET MULTILOCKS ON
CURSORSETPROP("Buffering", 5, "Customer")   && Enable table buffering
REPLACE caddress1 WITH "500 Water St." IN Customer  &&& Make a
change
lcUpdateXML = XMLUPDATEGRAM("Customer")  && Generate the Update Gram
STRTOFILE(lcUpdateXML, "UpdateGram.xml")
MODIFY FILE UpdateGram.xml
```

Die Downloaddateien auf www.hentzenwerke.com enthalten auch die in diesem Kapitel eingesetzte Tabelle Customer.

Die Datei UpdateGram.xml sollte den folgenden Inhalt haben:

```
<?xml version = "1.0" encoding="Windows-1252" standalone="yes"?>
<root xmlns:updg="urn:schemas-microsoft-com:xml-updategram">
      <updg:sync>
         <updg:before>
            <customer>
               <iid>1</iid>
               <cacctno>001000</cacctno>
               <cname>Journey Communications</cname>
               <caddress1>101 Main St.</caddress1>
               <ccity>Richmond</ccity>
               <cstate>VA</cstate>
               <czip>22901</czip>
            </customer>
         </updg:before>
         <updg:after>
            <customer>
               <iid>1</iid>
               <cacctno>001000</cacctno>
               <cname>Journey Communications</cname>
               <caddress1>500 Water St.</caddress1>
               <ccity>Richmond</ccity>
               <cstate>VA</cstate>
               <czip>22901</czip>
            </customer>
         </updg:after>
      </updg:sync>
</root>
```

Das Updategram beinhaltet einen einzelnen Knoten „root“, der nur das Element <updg:sync> enthält. In dieses Element eingebettet sind die Elemente <updg:before> und <updg:after>. Das Element <updg:before> enthält eine Liste der Originalwerte aller Felder des Datensatzes. Das Element <updg:after> enthält eine Liste aller aktuellen Werte. Beachten Sie, dass das Element <caddress1> den neuen Wert „500 Water St.“ enthält.

Wenn Sie mit CURSORSETPROP() die Schlüsselfeldliste setzen, bevor Sie XMLUpdateGram() ausführen, enthält das Updategram nur das Schlüsselfeld sowie die geänderten Felder. Das können Sie testen, indem Sie im Befehlsfenster die folgenden Zeilen eingeben:

```
TABLEREVERT(.T., "Customer")  && Revert the changes to the table
CURSORSETPROP("KeyFieldList", "iid", "Customer") && Set the key
field list
REPLACE caddress1 WITH "500 Water St." IN Customer  &&& Make a
change
lcUpdateXML = XMLUPDATEGRAM("Customer")  && Generate the Update Gram
STRTOFILE(lcUpdateXML, "UpdateGram1.xml")
MODIFY FILE UpdateGram1.xml
```

Die Datei UpdateGram1.xml sollte den folgenden Inhalt haben:

```
<?xml version = "1.0" encoding="Windows-1252" standalone="yes"?>

<root xmlns:updg="urn:schemas-microsoft-com:xml-updategram">
      <updg:sync>
         <updg:before>
            <customer>
               <iid>1</iid>
               <caddress1>101 Main St.</caddress1>
            </customer>
         </updg:before>
         <updg:after>
            <customer>
               <iid>1</iid>
               <caddress1>500 Water St.</caddress1>
            </customer>
         </updg:after>
      </updg:sync>
</root>
```

Beachten Sie, dass das Updategram nur die Informationen aus zwei Feldern enthält – das angegebene Schlüsselfeld (iid) und das geänderte Feld (caddress1).

Hinzufügen von Datensätzen

Um die Arbeitsweise von XMLUpdateGram() beim Hinzufügen von Datensätzen zu demonstrieren, geben Sie im Befehlsfenster das Folgende ein:

```
TABLEREVERT(.T., "Customer")  && Revert the changes to the table
INSERT INTO Customer (cacctno, cname, caddress1, ccity, cState,
czip);
      VALUES ("001004", "The Fox", "952 Market St.", "Reston", "VA",
"22903")
lcUpdateXML = XMLUPDATEGRAM("Customer")  && Generate the Update Gram
STRTOFILE(lcUpdateXML, "UpdateGram2.xml")
MODIFY FILE UpdateGram2.xml
```

Die Datei UpdateGram2.xml sollte den folgenden Inhalt haben:

```
<?xml version = "1.0" encoding="Windows-1252" standalone="yes"?>

<root xmlns:updg="urn:schemas-microsoft-com:xml-updategram">
      <updg:sync>
         <updg:before/>
         <updg:after>
            <customer>
               <iid>6</iid>
               <cacctno>001004</cacctno>
               <cname>The Fox</cname>
               <caddress1>952 Market St.</caddress1>
               <ccity>Reston</ccity>
```

```
            <cstate>VA</cstate>
            <czip>22903</czip>
         </customer>
      <updg:after>
   </updg:sync>
</root>
```

Beachten Sie, dass es sich bei dem Element <updg:before> um ein einzelnes leeres Tagelement handelt – da wir einen neuen Datensatz anlegen, gibt es keine „Vorher"-Daten. Das Element <updg:after> enthält alle Werte des neuen Datensatzes.

Löschen von Datensätzen

Um die Arbeitsweise von XMLUpdateGram() beim Löschen von Datensätzen zu demonstrieren geben Sie im Befehlsfenster das Folgende ein:

```
TABLEREVERT(.T., "Customer")  && Revert the changes to the table
LOCATE  && Move the record pointer to the first record
DELETE  && Delete the first record
lcUpdateXML = XMLUPDATEGRAM("Customer")  && Generate the Update Gram
STRTOFILE(lcUpdateXML, "UpdateGram3.xml")
MODIFY FILE UpdateGram3.xml
```

Die Datei UpdateGram3.xml sollte den folgenden Inhalt haben:

```
<?xml version = "1.0" encoding="Windows-1252" standalone="yes"?>

<root xmlns:updg="urn:schemas-microsoft-com:xml-updategram">
   <updg:sync>
      <updg:before>
         <customer>
            <iid>1</iid>
            <cacctno>001000</cacctno>
            <cname>Journey Communications</cname>
            <caddress1>101 Main St.</caddress1>
            <ccity>Richmond</ccity>
            <cstate>VA</cstate>
            <czip>22901</czip>
         </customer>
      </updg:before>
      <updg:after/>
   </updg:sync>
</root>
```

Prüfen auf Update-Konflikte

Warum schließt die Funktion XMLUpdateGram() beim Erzeugen des Updategrams auch die Originalwerte mit ein? Auf diese Weise können Sie überprüfen, dass in der Zwischenzeit kein anderer Anwender den geänderten oder gelöschten Datensatz geändert hat.

Die Benutzung von Updategrams im SQL Server 2000

Wo können Sie Updategrams einsetzen? Zunächst einmal sind Updategrams ein neues Feature des SQL Server 2000, mit dessen Hilfe in XML definierte Datenbankupdates durchgeführt werden können. Beachten Sie in den generierten Updategrams den Knoten „root“, der Namespace ist angegeben als:

```
<root xmlns:updg="urn:schemas-microsoft-com:xml-updategram">
```

Das für die mit VFP 7 genutzte Format für die Updategrams ist auch das Standardformat des SQL Server 2000.

Einsatz von Updategrams in verteilten Anwendungen

Auch wenn Sie den SQL Server 2000 nicht als Backend-Datenbank einsetzen, kann der Einsatz von Updategrams in mehrschichtigen Architekturen sinnvoll sein. Die Windows Distributed Internet Architecture (Windows DNA) legt fest, dass Geschäftsobjekte aus zwei Teilen zusammengesetzt sein sollen – einem beschreibenden und einem ausführenden. In einem verteilten Desktop-System befindet sich der beschreibende Teil des Geschäftsobjekts auf der Workstation, der ausführende Teil sitzt auf dem Anwendungsserver irgendwo im Netzwerk.

Wenn die Workstation Daten benötigt, sendet sie eine Anfrage an den beschreibenden Teil des Geschäftsobjekts. Statt die Daten direkt vom Backend zu empfangen, übergibt das Geschäftsobjekt die Anfrage an ein ausführendes Objekt. Das ausführende Objekt empfängt die Daten vom Backend und übergibt sie dem beschreibenden. Welchen Transportmechanismus kann dabei eingesetzt werden? In der Regel fällt die Wahl auf ADO und XML, aber die neuen XML-Fähigkeiten von VFP 7 zusammen mit der sehr schnellen Stringverarbeitung sind überzeugende Argumente für den Einsatz von XML.

Empfängt das Geschäftsobjekt auf der Workstation die Ergebnisdaten als XML-String kann es diese in einen Cursor von VFP umwandeln, der dann von der Desktop-Anwendung verarbeitet werden kann. Die Anwendung manipuliert den Cursor – ändert, erstellt und löscht Datensätze. Wenn der Anwender seine Arbeit speichern will, kann das Geschäftsobjekt XMLUpdateGram() gegen den Cursor ausführen und das daraus resultierende Updategram an den ausführenden Teil übergeben, der die Änderungen an die Backend-Datenbank weiterleitet.

VFPXMLProgID

Die neue Eigenschaft VFP.VFPXMLProgID ermöglicht Ihnen die Angabe einer COM-Komponente, die statt der nativen XML-Funktionen von VFP 7 (CursorToXML(), XMLToCursor(), XMLUpdateGram()) genutzt werden soll.

Die Eigenschaft enthält die programmatische Identifikation der ersetzenden COM-Komponente. Die von Ihnen angegebene Komponente muss die Schnittstelle IVFPXML von VFP 7 implementieren. Unter dem Eintrag „VFPXMLProgID-Eigenschaft" finden Sie in der Hilfedatei von VFP 7 ein Beispiel für die Erstellung eines COM-Objekts, das diese Schnittstelle enthält.

Einführung in die Webdienste

Webdienste sind die nächste Generation der Webanwendungen. Es handelt sich um modulare, selbstbeschreibende Anwendungen, die über das Web verteilt und aufgerufen werden können. Webdienste können von einfachen Funktionalitäten bis hin zu komplexen Geschäftsabläufen alles enthalten.

Eine einfache Möglichkeit zu verstehen, was Webdienste eigentlich sind, besteht darin, einen Blick auf die Arbeitsweise heutiger Webanwendungen zu werfen. Wenn Sie zur Zeit online ein Hotelzimmer buchen, müssen Sie in der Regel eine andere Website aufrufen um zu erfahren, wie das Wetter während Ihres Aufenthalts sein wird. Wäre es nicht besser, wenn die Website des Hotels diese Aufgabe für Sie übernehmen würde? Dafür ist es notwendig, dass die Website des Hotels in der Lage ist, mit der Website eines Wetterdienstes zu kommunizieren und die Wetterinformationen auf der Basis des Standorts des Hotels und Ihres Aufenthaltszeitraums an Sie weiterzuleiten.

Diese Funktionalität gibt es schon einige Zeit. Einige Websites von Hotels zeigen das aktuelle Wetter an; allerdings nutzen sie dafür in der Regel keine Webdienste, sondern haben dies „hart kodiert". In der Regel findet sich auf der Website eines Wetterdienstes eine GIF-Datei mit den Wetterdaten und die Site des Hotels (oder eines anderen Betreibers) enthält einfach einen Zeiger auf diese Datei. Diese Lösung ist nicht flexibel, da die Wetterinformation in einem bestimmten Anzeigeformat (Größe, Form, Farbe usw.) dargeboten werden muss. Der Einsatz von Webdiensten für den Bezug der aktuellen Wetterdaten ermöglicht es den Benutzern dieses Dienstes, die Information in jeder gewünschten Form anzuzeigen.

Bestehende Webdienste

Woher erfahren Sie, welche Webdienste verfügbar sind? Es gibt eine ganze Anzahl Ressourcen, in denen Sie suchen können. Eine davon ist http://www.xmethods.net/, die Ihnen eine Liste verfügbarer Webdienste zur Verfügung stellt (vgl. Tabelle 3).

Schon ein Blick auf diese Liste kann Ihnen helfen zu verstehen, wie Webdienste Webanwendungen revolutionieren können. Webdienste sind für Webanwendungen das gleiche wie ActiveX-Kontrollelemente für Win32-Anwendungen. Statt das Rad neu zu erfinden, können Sie die Dienste bestehender Websites in Anspruch nehmen, um Ihre Webanwendungen zu erweitern.

Tabelle 3. Einige der interessanteren Webdienste, die im Internet verfügbar sind.

Name	Beschreibung	URL
FedEx Tracker	Zugriff auf die Sendungsverfolgung von FexEs	http://www.xmethods.net/sd/2001/FedExTrackerService.wsdl
Babel Fish Service	Babel Fish Übersetzung	http://www.xmethods.net/sd/2001/BabelFishService.wsdl
Headline News	Neueste News aus unterschiedlichen Quellen	http://www.SoapClient.com/xml/SQLDataSoap.WSDL
Who Is	Eine SOAP-Version des Standard-„who is"-Dienstes, der Informationen über eine angegebene Internet-Domain liefert.	http://webservices.matlus.com/scripts/whoiswebservice.dll/wsdl/IWhoIs
Soap Web Search	SOAP-Schnittstelle zu den wichtigsten Suchmaschinen	http://www.SoapClient.com/xml/SQLDataSoap.WSDL
SMS Messaging	Senden Sie Textnachrichten an Mobil-Telefone	http://www.soapengine.com/lucin/soapenginex/smsx.asmx?wsdl
Currency Converter	Rechnet Werte einer internationalen Währung in eine andere um.	http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl

Zusätzlich zum Gebrauch von Webdiensten, die andere Sites zur Verfügung stellen, können Sie auch Ihre eigenen VFP 7 COM-Server als Webdienste zur Verfügung stellen, so dass andere darauf zugreifen.

Simple Object Access Protocol (SOAP)

Denkt man über das Beispiel der Website des Hotels nach, das mit einer Website kommuniziert, die Wetterinformationen zur Verfügung stellt, kommen einige interessante Fragen auf – woher weiß die Site des Hotels, welche Möglichkeiten die Wetterseite bietet? Woher erfährt sie, in welchem Format die Anfrage gesendet werden muss und in welchem Format die Antwort zurückkommt?

Die Antwort auf diese Fragen heißt SOAP – Simple Object Access Protocol. SOAP benutzt die bestehenden Standards von HTTP und XML, um eine genormte Möglichkeit zur Verfügung zu stellen, mit der Befehle und Parameter zwischen den Clients und dem Server ausgetauscht werden können. Die Clients senden ihre Anfragen im XML-Format an einen Server und der Server sendet die Ergebnisse als XML zurück. Dieser Datenverkehr wird über HTTP abgewickelt.

Nähere Informationen über Themen, die mit SOAP in Verbindung stehen, finden Sie auf Microsofts Website http://msdn.microsoft.com/soap/.

Einsatz des SOAP Toolkit 2.0

Um die Veröffentlichung von Webdiensten und die Erstellung von Anwendungen, die Webdienste über SOAP aufrufen, zu vereinfachen, hat Microsoft das SOAP Toolkit (als dieser Text geschrieben wurde war die Version 2.0 aktuell) entwickelt.

Das SOAP Toolkit wird mit einer exzellenten Hilfedatei (Soap.CHM) ausgeliefert, die Sie im Verzeichnis MSSOAP/Documentation finden. Statt diese Informationen hier zu wiederholen, befassen wir uns lieber mit den Highlights des Toolkit. Details erfahren Sie in der Hilfedatei und in den Beispielen.

Installieren des SOAP Toolkit

Sie können das SOAP Toolkit von http://msdn.microsoft.com/soap/ herunterladen. Der Download besteht aus zwei Dateien: dem SQAP Toolkit 2.0 Gold Release (SoapToolkit20.EXE) und Gold Release Beispielen (SoapToolkit20Samples.EXE).

Nach dem Herunterladen des Toolkit klicken Sie doppelt auf die EXE-Dateien und folgen den Anweisungen des Installationsassistenten. Als Vorgabe wird das SOAP Toolkit im Verzeichnis Programme\MMSOAP installiert.

Das Toolkit enthält die folgenden Komponenten und Werkzeuge:

- Eine Komponente, die es Clients vereinfacht, auf Webdienste zuzugreifen.
- Eine Komponente, die sich auf dem Webserver befindet und Operationen der Webdienste in Methodenaufrufe von COM-Objekten umwandelt.
- Zusätzliche Komponenten für die Erstellung, Übermittlung, das Lesen und Ausführen von SOAP-Nachrichten.
- Ein Werkzeug, dass Dokumente in der Web Services Description Language (WDSL) und der Web Services Meta Language generiert (vgl. die entsprechenden folgenden Abschnitte).
- Das Framework SOAP Messaging Object (SMO), das in SOAP-Nachrichten eine Alternative zum XML DOM darstellt.

Web Services Description Language (WSDL)

Die Web Services Description Language (WSDL) kann mit einer Typbibliothek von XML verglichen werden. Diese Datei identifiziert die Dienste und Verarbeitungsschritte, die durch einen Server bereitgestellt werden. Die WSDL-Datei befindet sich auf dem Server. Jeder Client, der die Dienste des Servers in Anspruch nehmen will, muss sich eine Kopie der Datei herunterladen, um entscheiden zu können, die die SOAP-Anfrage für diesen Provider formatiert werden muss.

Dokumentenorientierte und RPC-orientierte Operationen

In einer WSDL-Datei werden zwei Haupttypen an Operationen definiert – dokumentenorientierte Operationen und RPC-orientierte Operationen (Remote Procedure Call). Bei dokumentenorientierten Operationen werden Anfragen und Antworten als XML-Dokumente versandt. Mit dem SOAP Messaging Object (SMO) ist es einfach, XML-Dokumente als SOAP-Nachricht auszuführen.

RPC-orientierte Operationen erwarten Eingabeparameter für die Anfrage und geben das Ergebnis als Rückgabewert zurück.

Web Services Meta Language (WSML)

Eine WSML-Datei (Web Services Meta Language) enthält Informationen, die eine Operation eines Dienstes, die in der WSDL-Datei beschrieben sind, auf eine Methode in einem COM-Objekt umlegt.

SOAP Listener

Ein Listener ist die Einheit, die hereinkommende SOAP-Nachrichten auf der Serverseite behandelt. Sie können zwischen zwei Arten von Listenern wählen:

- Einem Internet Server API (ISAPI)-Listener
- Einem Active Server Pages (ASP)-Listener

Aus der Perspektive der Performance ist ein ISAPI-Listener die erste Wahl. Teilweise ist er um den Faktor vier schneller. Wenn Sie aber vorhaben, zwischen der Anfrage des Client und dem Methodenaufruf des COM-Servers noch einige Verarbeitungsschritte zwischenzuschalten, sollten Sie einen ASP-Listener in Erwägung ziehen. Sie können innerhalb einer ASP-Seite Code platzieren. Vergleichen Sie zu diesem Thema in der Hilfedatei des SOAP Toolkit die Einträge „Specifying an ASP Listener“ und „Specifying an ISAPI Listener“.

Die Webdienst-Erweiterungen von Visual FoxPro 7.0

Obwohl Sie das Microsoft SOAP Toolkit 2.0 auch direkt ansprechen können, enthält Visual FoxPro 7.0 Erweiterungen für das Toolkit, die es einfacher machen, einen Webdienst für den Gebrauch durch den Client zu registrieren (wenn Sie eine genauere Kontrolle über die Ein- und Ausgaben von Webdiensten benötigen, sollten Sie das SOAP Toolkit allerdings direkt ansprechen). Die Erweiterungen erleichtern es auch, Webdienste zu erstellen und können die WSDL- und WSML-Dateien für Ihre COM-Server erstellen.

Genaue Informationen über das SOAP Toolkit mit Visual FoxPro erhalten Sie in Rick Strahls Beschreibung „Using Microsoft SOAP Toolkit for remote object access“ auf http://www.west-wind.com/presentations/soap/.

Registrieren eines Webdienstes

Um einen Webdienst für den Gebrauch durch Clients zu registrieren, führen Sie die folgenden Schritte aus:

1. Rufen Sie im Menü Extras den IntelliSense-Manager auf.
2. Wechseln Sie zur Seite Typen und klicken auf die Schaltfläche Webdienste (vgl. Abbildung 1)

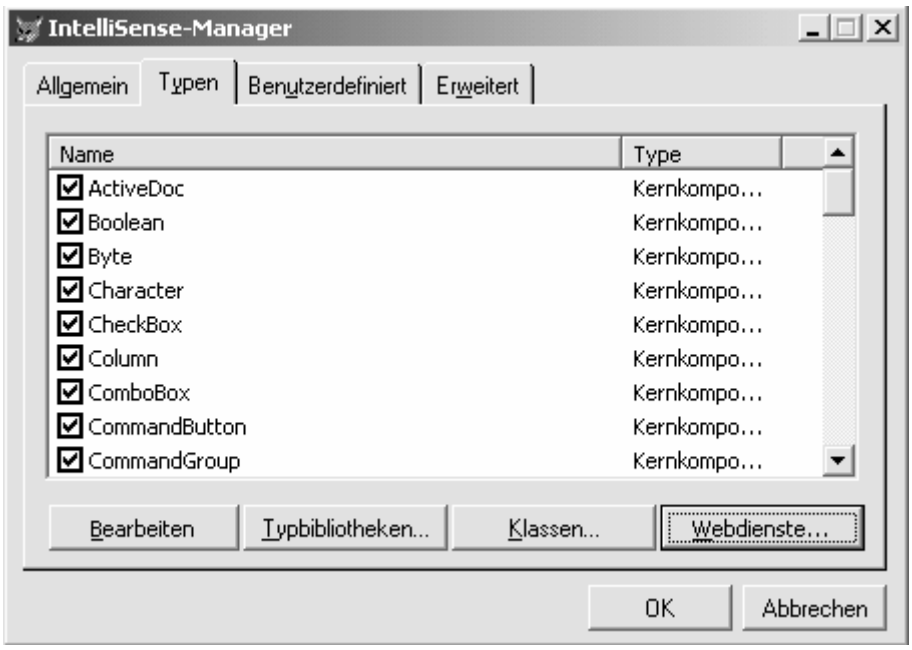

Abbildung 1. Die Schaltfläche Webdienste des IntelliSense-Managers ruft den Dialog für die Registrierung von Webdiensten auf.

3. Im Dialog „Registrierung der Visual FoxPro Webdienste“ geben Sie einen netten Namen für den Webdienst ein (siehe Abbildung 2). Der Name kann auch Leerzeichen enthalten. Eine Historie aller Webdienste, die Sie bereits besucht haben, wird in der Combobox Webdienstname angezeigt.

Abbildung 2. Im Dialog "Registrierung der Visual FoxPro Webdienste" können Sie einen Webdienst registrieren, der sich irgendwo im Internet befindet und den Sie in Ihren Anwendungen nutzen wollen.

4. Im Feld WDSL URL geben Sie die WSDL URL des Webdienstes ein, den Sie registrieren. Ein Beispiel: http://www.xmethods.net/sd/ BabelFish-Service.wsdl.
5. Klicken Sie auf die Schaltfläche Registrieren.

Jetzt generiert der Dialog auf der Basis der von Ihnen eingegebenen Informationen IntelliSense-Skripts. Beachten Sie, dass dies einige Sekunden in Anspruch nehmen kann, ohne dass Sie eine Rückmeldung über den Fortschritt der Verarbeitung erhalten. Ist der Prozess beendet, erhalten Sie die Meldung, die Sie in Abbildung 3 sehen.

Abbildung 3. Dieser Dialog zeigt Ihnen an, dass die Registrierung des Webdienstes erfolgreich abgeschlossen wurde. Beachten Sie, dass es eine Weile dauern kann, bis dieser Dialog erscheint.

Schließen Sie den Dialog mit einem Klick auf OK. Ihr neuer Webdienst wird der Liste auf der Seite Typen des IntelliSense-Managers hinzugefügt. Im nächsten Abschnitt behandeln wir den Zugriff auf Ihren neuen Webdienst.

Es ist nicht erforderlich, dass Sie den Webdienst im IntelliSense-Manager registrieren, aber wenn Sie die Registrierung vorgenommen haben, bietet VFP Ihnen IntelliSense für die Methoden und Parameter des Webdienstes.

Der Aufruf der Webdienste

Um einen Webdienst programmatisch aufzurufen, folgen Sie diesen Schritten:

1. Deklarieren Sie im Code eine typgebundene Variable. Ein Beispiel:

```
LOCAL loTranslate AS
```

2. Wählen sie in der Dropdownliste der Typen von IntelliSense, wählen Sie einen Webdienst. Ein Beispiel:

```
LOCAL loTranslate AS Language Translation
```

3. Wenn Sie Enter drücken, wird der Proxycode des Webdienstes automatisch Ihrer Quelldatei hinzugefügt. Ein Beispiel:

```
LOCAL loTranslate AS Language Translation
LOCAL loWS
loWS = NEWOBJECT("Wsclient",HOME()+"ffc\_webservices.vcx")
loTranslate =
loWS.SetupClient("http://www.methods.net/sd/BabelFishService.wsdl
","BabelFish", "BabelFishPort")
```

Dieser Code instanziiert die Basisklasse WSClient (loWS) von VFP 7 und erstellt anschließend ein SOAP-Clientobjekt (loTranslate).

4. Um diesen Dienst zu nutzen, fügen Sie den folgenden Code hinzu:

```
?loWS.BabelFish("en_es","fox"
```

Dieser Code teilt dem Webdienst Babelfish mit, dass er das Wort „fox“ vom Englischen ins Spanische („en_es“) übersetzen soll und zeigt das Ergebnis auf dem Desktop von VFP an.

Veröffentlichen von Webdiensten

Zusätzlich zum Aufruf der Webdienste ermöglichen die VFP 7 Web Service-Erweiterungen die Veröffentlichung Ihrer mit VFP entwickelten COM-Server als Webdienst.

Um Webdienste anbieten zu können, müssen Sie den Internet Information Server installiert haben.

Um einen Webdienst anzubieten, führen Sie die folgenden Schritte aus:

1. Erstellen Sie ein COM Server-Projekt, das mindestens eine OLE Public-Klasse enthält und erstellen Sie daraus eine COM-Server-DLL (in der Regel eine Multi-Thread-COM-DLL).
2. Klicken Sie mit der rechten Maustaste auf das Projekt mit dem COM-Server und wählen aus dem Kontextmenü „Generator“. Damit rufen Sie die Assistentenauswahl auf. Dort wählen Sie den Webdiensteverleger und klicken auf OK.
3. Wenn Sie das erste Mal einen Webdienst veröffentlichen, erscheint der Dialog, den Sie in Abbildung 5 sehen und der Ihnen empfiehlt, eine URL für Ihre Dateien zur Webdienstunterstützung auszuwählen, also für die ASP Listeners sowie die WSDL- und WSML-Dateien. Das Vorgabeverzeichnis ist zwar nicht unbedingt erforderlich, aber wenn Sie keines angeben, werden Sie jedes Mal, wenn Sie einen Webdienst veröffentlichen, nach dem Verzeichnis zum Speichern gefragt. Klicken Sie auf OK, um den Dialog „Visual FoxPro Webdienst-Speicherort“ aufzurufen.

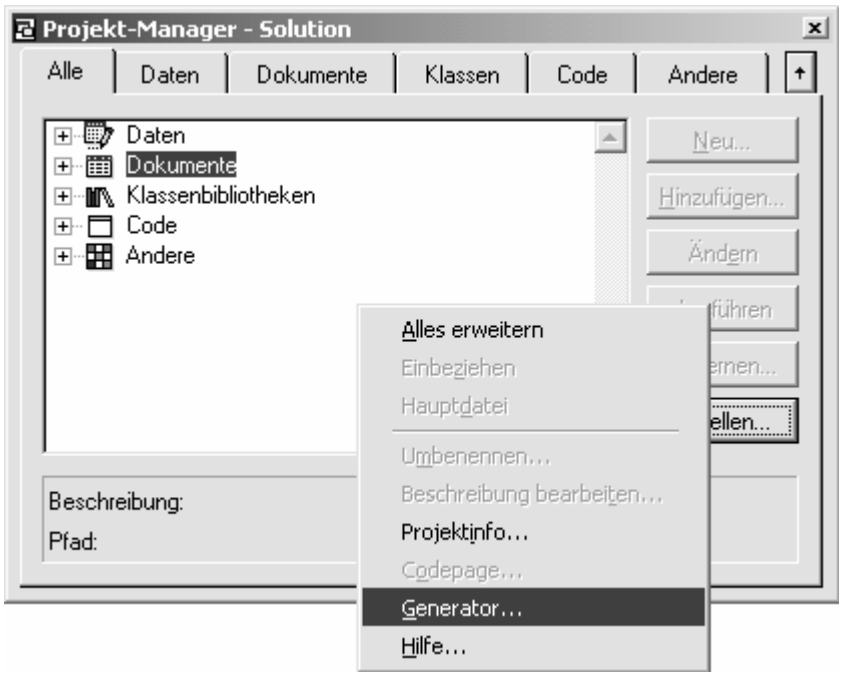

Abbildung 4. Sie können den Dialog für die Auswahl der Generatoren im Kontextmenü des Projektmanagers aufrufen. Alternativ dazu können Sie auch im Menü von VFP 7 Extras - Assistenten - Alle Assistenten aufrufen und im dort erscheinenden Dialog Webdiensteverleger auswählen.

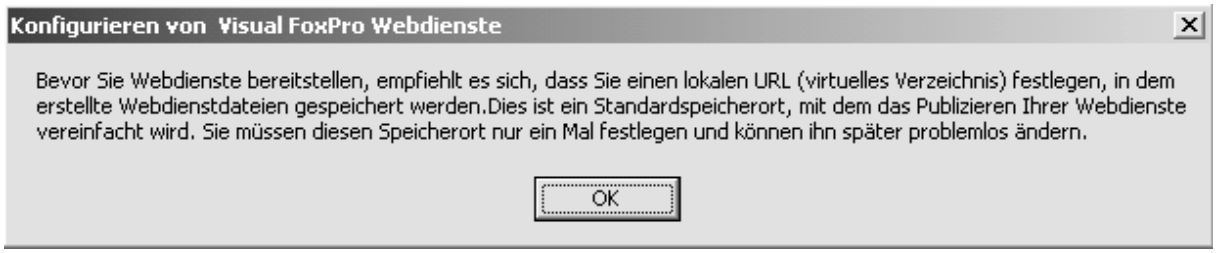

Abbildung 5. Wenn Sie zum ersten Mal einen Webdienst veröffentlichen, empfiehlt Ihnen der Webdienstkonfigurationsdialog, dass Sie eine Standard-URL für Ihre Dateien der Webdienste auswählen.

1. In diesem Dialog wählen Sie ein bestehendes virtuelles Verzeichnis für Ihre Dateien oder erstellen ein neues (vgl. Abbildung 6).

   Sie können ein bestehendes virtuelles Verzeichnis durch einen Klick auf die Option „Vorhanden“ auswählen und in der Combobox „Virtuelles Verzeichnis“ ein Verzeichnis wählen.

   Ein neues Verzeichnis legen Sie durch Anklicken der Option „Neu“ und die Eingabe eines Verzeichnisnamens in die Box „Neuer Name des virtuellen Verzeichnisses“ an. In der Regel sollten Sie ein neues Verzeichnis unterhalb Ihres Verzeichnisses c:\inetpub\wwwroot anlegen.

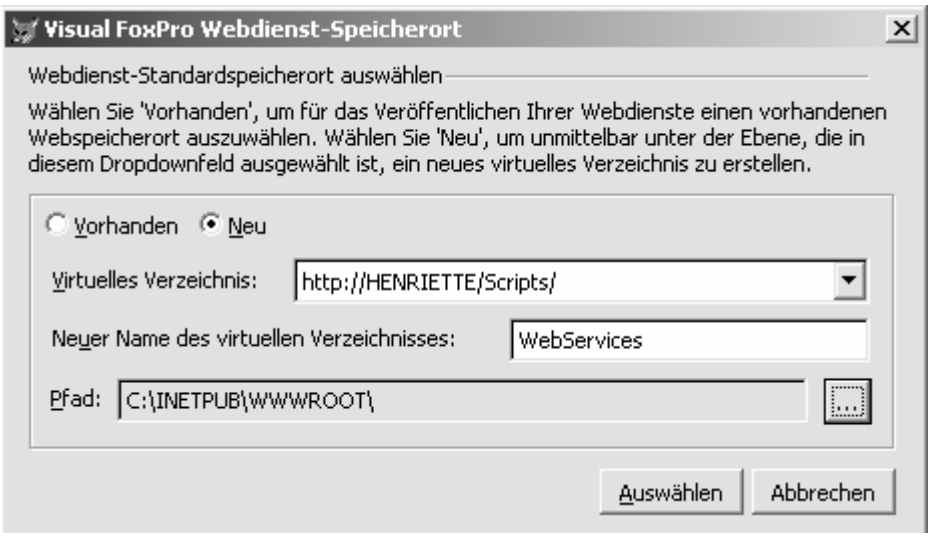

Abbildung 6. Der Dialog Webdienst-Speicherort ermöglicht Ihnen die Angabe eines vorgegebenen virtuellen Verzeichnisses für Ihre Unterstützungsdateien für Webdienste.

Die im Dialog eingegebenen Informationen werden in einer Tabelle mit Namen FoxWS.DBF gespeichert, die sich in Ihrem Programmverzeichnis von Visual FoxPro 7 befindet.

2. Klicken Sie auf die Schaltfläche „Auswählen“, um mit der Arbeit fortzufahren. Damit rufen Sie den Dialog Webdienste veröffentlichen auf, den Sie in Abbildung z sehen. Als Vorgabewert enthält die Combobox COM-Server sowohl alle COM-Server, die zu den geöffneten Projekten gehören als auch die vorher veröffentlichten COM-Server. Wählen Sie einen COM-Server in der Liste oder klicken auf die Schaltfläche, um einen COM-Server auszuwählen, der sich nicht in der Liste befindet.
3. Die Combobox Select Class zeigt eine Liste der Klassen, die im ausgewählten COM-Server als public gekennzeichnet sind. Sollte der von Ihnen gewählte COM-Server nur eine Klasse beinhalten ist dieses Feld deaktiviert.

   Wenn die von Ihnen gewählte Klasse Währungs- oder Variant-Parameter enthält oder Werte zurückliefert, ist eine Information-Grafik neben der Combobox Select Class angezeigt (vgl. dazu Abbildung 7). Klicken Sie auf diese Grafik, teilt Ihnen ein Dialog mit, dass Sie für diese Klasse eine typgebundene Variable definieren müssen, andernfalls würde eine ungültige WSDL-Datei generiert.

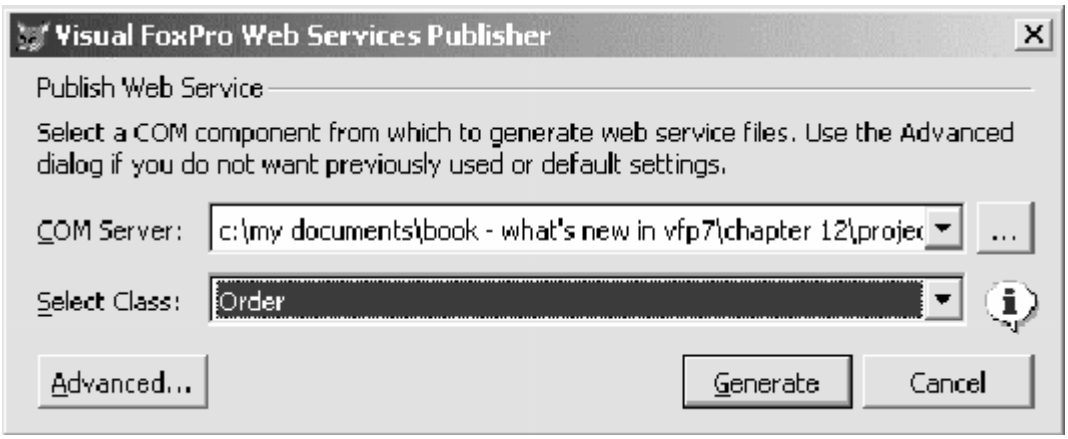

Abbildung 7. Der Web Services Publisher ermöglicht Ihnen die Generierung der Hilfsdateien für eine COM-Komponenten von VFP 7.

4. Wenn Sie auf die Schaltfläche „Erweitert“ klicken, wird der in Abbildung 8 gezeigte Dialog aufgerufen. Hier können Sie zusätzliche Informationen eingeben, z. B.:

   - Die URL und den Namen der WSDL-Dateien.
   - Den Typen des Listeners (ISAPI oder ASP)
   - Wenn Sie einen ASP-Listener einsetzen, Informationen über die ASP-Dateien.
   - IntelliSense-Skripts

   Detaillierte Informationen über diese Einstellungen finden Sie in der Hilfedatei von VFP 7 unter dem Eintrag „Webpublishing-Assistent“.

Wenn der Webpublishing-Assistent erfolgreich beendet wurde, erscheint ein Dialog, der dem in Abbildung 9 gezeigten entspricht.

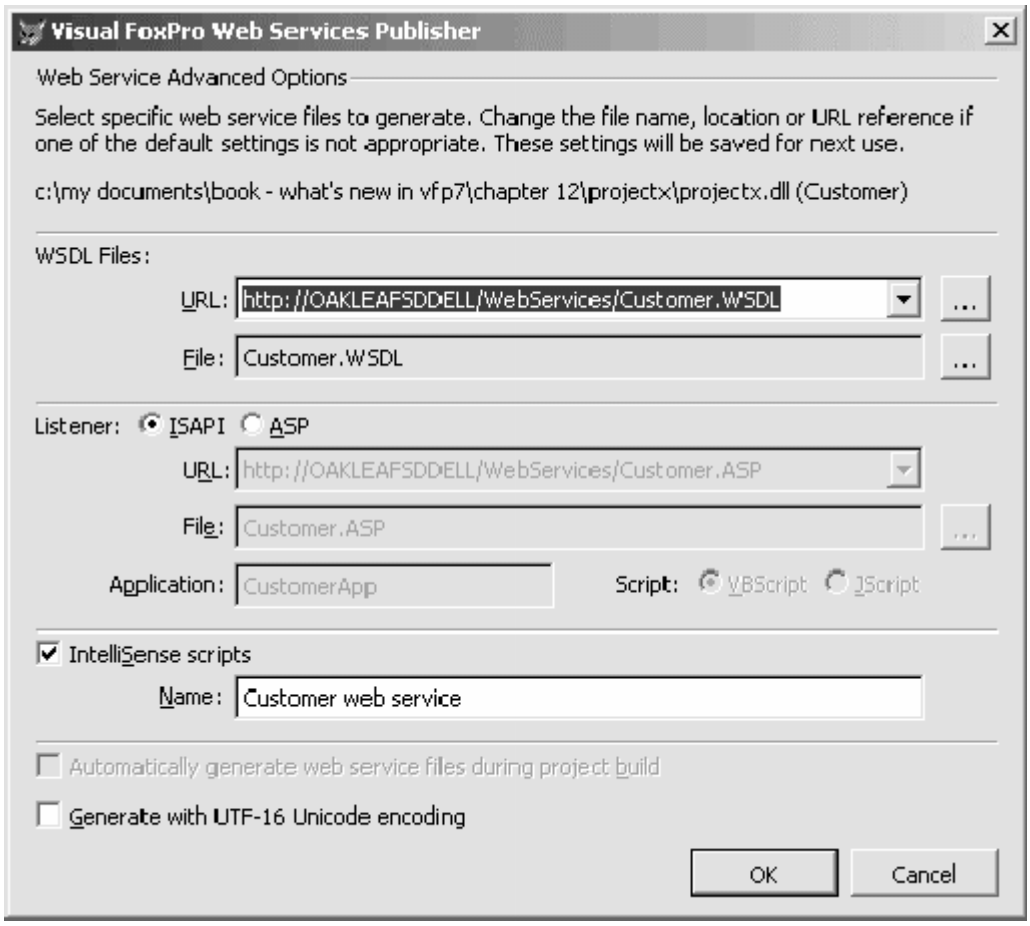

Abbildung 8. Der erweiterte Dialog des Webpublishing-Assistenten ermöglicht Ihnen die Angabe zusätzlicher Informationen über die generierten Hilfsdateine.

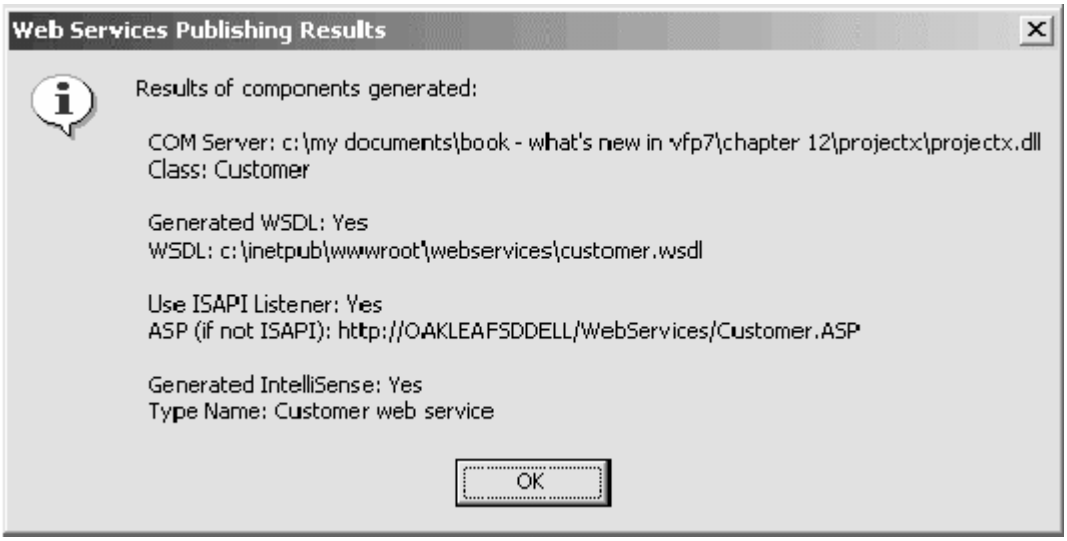

Abbildung 9. Nach erfolgreichem Erstellen der Hilfsdateien für Ihre COM-Komponente zeigt der Webpublishing-Assistent diesen Dialog an.

Das wichtigste Nebenprodukt des Veröffentlichungsprozesses sind die WSDL- und WSML-Dateien. Diese Dateien werden in dem Verzeichnis veröffentlicht, das Sie im Dialog des Webpublishing-Assistenten angegeben haben. Hier ein Teil des Listings einer WSDL-Datei, die von der benutzerdefi-

nierten Klasse ProjectX generiert wurde. Mit Ausnahme von GetCustomerByAcctNo sind alle Methoden geschützt:

```
<?xml version ='1.0' encoding='UTF-8' ?>
     <!-- Generated 04/27/01 by Microsoft SOAP Toolkit WSDL File
Generator, Version 1.00.623.0 -->
<definitions name='customer' targetNameSpace =
'http://tempuri.org/wsdl/'
        ...
     <message name='Customer.GetCustomerByAcctNo'>
        <part name='AcctNo' type='xsd:string'/>
     </message>
     <message name='Customer.GetCustomerByAcctNoResponse'>
        <part name='Result' type='xsd:string'/>
     </message>
     <portType name='CustomerSoapPort'>
        <operation name='GetCustomerByAcctNo'
parameterOrder='AcctNo'>
           <input message='wsdlns:Customer.GetCustomerByAcctNo'/>
           <output
message='wsdlns:Customer.GetCustomerByAcctNoResponse' />
        </operation>
     </portType>
     <binding name='CustomerSoapBinding'
type='wsdlns:CustomerSoapPort' >
        <stk:binding preferredEncoding='UTF-8'/>
        <soap:binding style='rpc'
transport='http://schemas.xmlsoap.org/soap/http'/>
        <operation name='GetCustomerByAcctNo' >
           <soap:operation
soapAction='http://tempuri.org/action/Customer.GetCustomerByAcctNo'
/>
           <input>
              <soap:body use='encoded'
namespace='http://tempuri.org/message/'
encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
           </input>
           <output>
              <soap:body use='encoded'
namespace='http://tempuri.org/message/'
encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
           </output>
        </operation>
     </binding>
     <service name='customer' >
        <port name='CusotmerSoapPort'
binding='wsdlns:CustomerSoapBinding' >
           <soap:address
location='http://OAKLEAFSDDELL/WebServices/customer.wsdl' />
        </port>
     </service>
</definitions>
```

Die Klasse, die wir in diesem Beispiel einsetzen, finden Sie im Downloadbereich unter www.hentzenwerke.com.

Hier die dazugehörende WSML-Datei, die durch den Webpublishing-Assistenten generiert wurde:

```
<?xml version ='1.0' encoding='UTF-8' ?>
     <!-- Generated 04/27/01 by Microsoft SOAP Toolkit WSDL File
Generator, Version 1.00.623.0 -->
<servicemapping name='customer'>
     <service name='customer'>
        <using PROGID='projectx.Cusomer' cachable='0'
ID='CusomerObject' />
        <port name='CustomerSoapPort'>
           <operation name='GetCustomerByAcctNo'>
              <execute uses='CustomerObject'
method='GetCustomerByAcctNo' dispID='0'>
                 <parameter callIndex='1' name='AcctNo'
elementName='AcctNo' />
                 <parameter callIndex='-1' name='retval'
elementName='Result' />
              </execute>
           </operation>
        </port>
     </service>
</servicemapping>
```

Eine Erklärung der Inhalte dieser Dateien finden Sie in der Hilfedatei des SOAP Toolkit 2.0.

Wenn während der Generierung der Dateien ein Fehler auftritt erscheint der in Abbildung 10 gezeigte Dialog. Sie können anschließend die WSDL-Datei prüfen, um die Parameter und/oder Rückgabewerte zu ermitteln, die nicht dargestellt werden konnten.

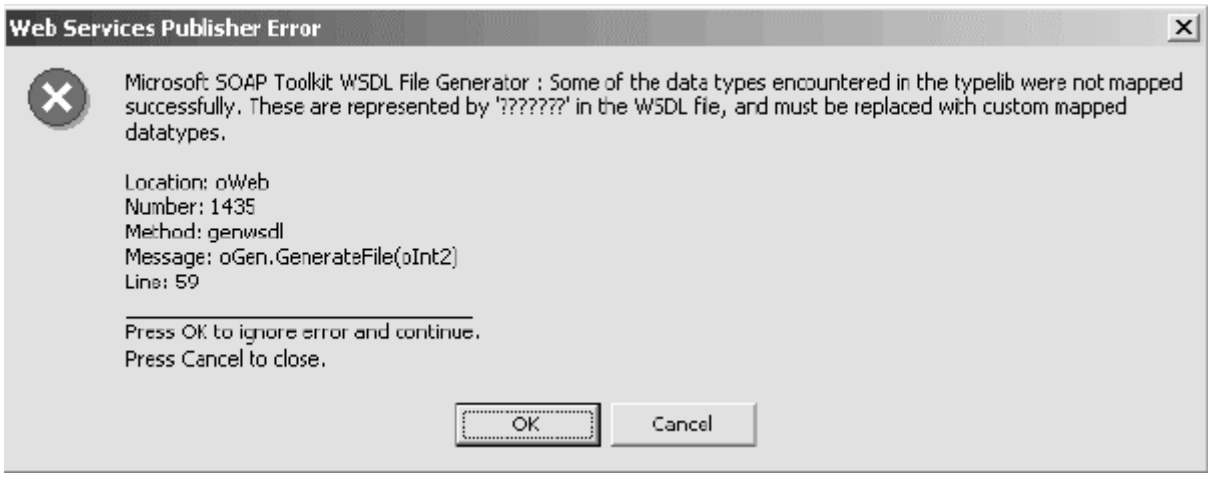

Abbildung 10. Der Webpublishing-Assistent teilt Ihnen mit, wenn Fehler während der Generierung der Hilfsdateien für Ihren COM-Komponente aufgetreten sind.

Der Projecthook Webdienste

Der Dialog „Erweitert“ des Webpublishing-Assistenten (Abbildung 8) enthält eine Checkbox, die es Ihnen ermöglicht, die Hilfsdateien automatisch während des Erstellens eines Projekts zu generieren. Wenn Sie diese Box aktivieren, wird ein Projecthook für Ihr Projekt erstellt. Dieser Hook ruft jedes Mal den Generator für die Hilfsdateien auf, wenn Sie Ihr Projekt neu erstellen.

Zusätzlich beendet der Hook automatisch alle IIS-Prozesse, die Ihren COM-Server sperren. Wenn Sie den Hook nicht einsetzen, müssen Sie diese Prozesse manuell beenden.

Zusammenfassung

Auch hier hat das FoxPro-Team gute Arbeit geleistet, um Visual FoxPro auch für zukünftige Anforderungen in der Softwareentwicklung fit zu machen. Die Erweiterungen in Visual FoxPro im Hinblick auf XML und Webdienste erleichtert die Erstellung von verteilten webbasierten Anwendungen der nächsten Generation.