Re: Execute stored procedure via web method.

Source:
http://www.tech−archive.net/Archive/DotNet/microsoft.public.dotnet.framework.webservices/2004−12/0080.html

---

From: Dan Rogers (*danro_at_microsoft.com*)
Date: 12/02/04

Date: Thu, 02 Dec 2004 22:24:56 GMT

Hi,

This is a great example. Russel, well done.

The only thing I might add is that you can generate a class named Command from this schema that lets you bypass any need to directly manipulate XML in your code at all. Using a tool like XSD.exe or XmlObjectGen.exe, you can read that schema and generate an assembly. Then you can change the web method to the following:

<codeSnippet language="C#">
 [WebMethod]
 public DataSet ExecuteCommand(command commandParam)

And then you can simpify the rest of the sample by just eliminating all of the Xml element manipulation and directly access the named members of the command class.

e.g. (assuming using XsdObjectGen.exe)

```
    SqlCommand command = new SqlCommand(commandParam.text);
     SqlParameterCollection commandParameters = command.Parameters;

   foreach (parameter p in commandParam)
   {

         SqlParameter parameter = new SqlParameter();
         parameter.ParameterName = p.name;
/// etc etc etc

   }
```

Also instead of parsing the qname (which would have to be something like "xs:int", "xs:string", etc), you can change the schema to make the type element a string as well, and then directly switch on the CLR type name (or whatever you wish to use as an enumerator of supported type names)

−−−−−−−−−−−−−−−−−−−

From: Drew Marsh <drub0y@hotmail.no.spamming.com>
Subject: Re: Execute stored procedure via web method.
References: <ubU9tTo0EHA.3808@tk2msftngp13.phx.gbl>
Content−Type: text/plain; charset=iso−8859−1
X−Newsreader: JetBrains Omea Reader 341.19
Message−ID: <O7l8qLp0EHA.3364@TK2MSFTNGP12.phx.gbl>
Newsgroups: microsoft.public.dotnet.framework.webservices
Date: Wed, 24 Nov 2004 18:03:10 −0800
NNTP−Posting−Host: 65.223.252.240
Lines: 1
Path:
cpmsftngxa10.phx.gbl!TK2MSFTFEED01.phx.gbl!TK2MSFTNGP08.phx.gbl!TK2MSFTNGP12
phx.gbl
Xref: cpmsftngxa10.phx.gbl
microsoft.public.dotnet.framework.webservices:7633
X−Tomcat−NG: microsoft.public.dotnet.framework.webservices

Russell Verdun wrote:

> Yes, I did get your answer, but not sure where to start, I'm not
> versed in XML, but it looks straight forward, where would XML code go?

Ahh, well we can help you out there. Unfortunately the answer to your
question is a lot longer than could possibly be explained and understood in
a newsgroup posting. I highly suggest reading up on XML, XML Schema and
WSDL. However, I'll explain how I would do it from a pure .NET perspective
and hopefully you can run with it from there.

I would suggest something like the following...

An schema like:

```
<xs:schema id="SimpleCommandExample"
targetNamespace="uri:marsh−samples−SimpleCommand"
elementFormDefault="qualified"
      xmlns="uri:marsh−samples−SimpleCommand"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
      attributeFormDefault="unqualified">
      <xs:complexType name="SimpleCommand">
            <xs:sequence>
                  <xs:element name="parameters" minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                              <xs:sequence>
                                    <xs:sequence>
                                          <xs:element name="parameter" type="SimpleCommandParameter"
minOccurs="0" maxOccurs="unbounded" />
                                    </xs:sequence>
                              </xs:sequence>
                        </xs:complexType>
                  </xs:element>
```

```
            </xs:sequence>
            <xs:attribute name="text" type="xs:string" />
        </xs:complexType>
        <xs:complexType name="SimpleCommandParameter">
            <xs:attribute name="name" type="xs:string" />
            <xs:attribute name="value" type="xs:string" />
            <xs:attribute name="type" type="xs:QName" />
        </xs:complexType>
        <xs:element name="command" type="SimpleCommand"></xs:element>
</xs:schema>
```

Which would result in instance documents like:

```
<command xmlns="uri:marsh−samples−SimpleCommand"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" text="SELECT * FROM MyTable
WHERE myField = @myParameter">
        <parameters>
            <parameter name="@myParameter" type="xsd:int" value="1" />
        </parameters>
</command>
```

And a web method that handles such commands like:

```
<codeSnippet language="C#">
  [WebMethod]
  [SoapDocumentMethod(ParameterStyle=SoapParameterStyle.Bare,
Use=SoapBindingUse.Literal)]
  [return:XmlElement("results",
Namespace="uri:marsh−samples−MyServiceTypes")]
  public XmlDataDocument ExecuteCommand([XmlElement("command",
Namespace="uri:marsh−samples−SimpleCommand")] XmlElement commandElement)
  {
    // TODO: run a validating reader over the element if you're concerned
someone might pass you invalid instances

    SqlCommand command = new
SqlCommand(commandElement.GetAttribute("text"));

    // NOTE: assumes schema is in default namespace which it probably
wouldn't be, so make sure to use overload where
    // XmlNamespaceManager is passed and use namespace prefix on XPath
elements
    XmlNodeList parameterElements =
commandElement.SelectNodes("parameters/parameter");

    if(parameterElements.Count > 0)
    {
        SqlParameterCollection commandParameters = command.Parameters;

        foreach(XmlElement parameterElement in parameterElements)
        {
```

```
        string xsdNamespacePrefix =
commandElement.OwnerDocument.GetPrefixOfNamespace("http://www.w3.org/2001/XM
LSchema");

        SqlParameter parameter = new SqlParameter();
        parameter.ParameterName = parameterElement.GetAttribute("name");

        string parameterElementValue =
parameterElement.GetAttribute("value");

        string parameterXsdType = parameterElement.GetAttribute("type");
        string[] qNameParts = parameterXsdType.Split(':');

        if(qNameParts[0] != xsdNamespacePrefix)
        {
          throw new NotSupportedException("Expecting parameter types
only from the Xml Schema Definition namespace.");
        }

        object parameterValue;

        switch(qNameParts[1])
        {
          case "int":
          case "nonNegativeInteger":
            parameterValue = XmlConvert.ToInt32(parameterElementValue);

            break;

          case "string":
            parameterValue = parameterElementValue;

            break;

          // TODO: handle all the XSD types you want support

          default:
            throw new NotSupportedException(string.Format("Unsupported
XSD type specified: {0}.", parameterXsdType));
        }

        parameter.Value = parameterValue;
      }
    }

    // TODO: execute the command and get back a real, filled dataset
    DataSet resultDataSet = new DataSet();

    XmlDataDocument result = new XmlDataDocument(resultDataSet);
```

```
    return result;
}
</codeSnippet>
```

HTH,
Drew