Informatiehuis Water

Een samenwerkingsverband van:

Interprovinciaal Overleg ipo

Rijkswaterstaat
Ministerie van Infrastructuur en Milieu

HetWaterschapshuis

Aquo Domeintabellen Services (Aquo DS)

Handleiding Webservice

handleiding voor de programmeur

Aquo

Auteur: Informatiehuis Water

Datum: 1 december 2011

Versie: 1.2

Documentbeheer

Wijzigingshistorie

Datum	Versie	Auteur	Wijziging
		Aat van den Heuvel (MX.Systems)	Initiële versie
4 aug. 2010	1.01	Aat van den Heuvel (MX.Systems)	Diverse aanpassingen o.a. wijziging in het datacontract van de web-service
26 sept. 2011	1.1	H.T. Reitsma (IHW)	IHW huisstijl toegepast URL's bijgewerkt
1 dec 2011	1.2	S. van Kuijck (IHW)	Tekstuele wijziging doorgevoerd in paragraaf 1.2

Review

Datum	Versie	Reviewer	Functie
20100804	1.01	Aat van den Heuvel (MX.Systems)	Diverse aanpassingen o.a. wijziging in het datacontract van de web-service
12-08-2010	1.01	R. Wigmans (MX.Systems)	Verwijzing naar WSDL opgenomen; (voorbeeld) output opgenomen

Controle en vrijgave

Datum	Versie	Controleur	Functie

Literatuurbronnen

- Functioneel Ontwerp Domeintabellensysteem, versie 1.4.4 (MX kenmerk P4796-R-1)

Inhoudsopgave

1. Toepassen webservice

1.1 Inleiding

Het doel van dit document is een handleiding te geven aan de programmeur bij het gebruik van de webservice van de Aquo Domeintabellen Services (Aquo DS). Het gebruik is beschreven voor een .NET-applicatie in Microsoft Visual Studio. Andere client-programmeervoorbeelden zoals Java of Perl zijn niet gedocumenteerd. De aansluitvoorwaarden van de webservice zijn vastgelegd door middel van de WSDL. Tevens is hiermee de input en output beschreven.

De stappen bij het beschikbaar maken van de webservice in een project zijn:

- Voeg de webservice aan de "References" toe
- Wijzig en controleer de gegenereerde *app.config*

1.2 Voeg webservice aan "References" toe

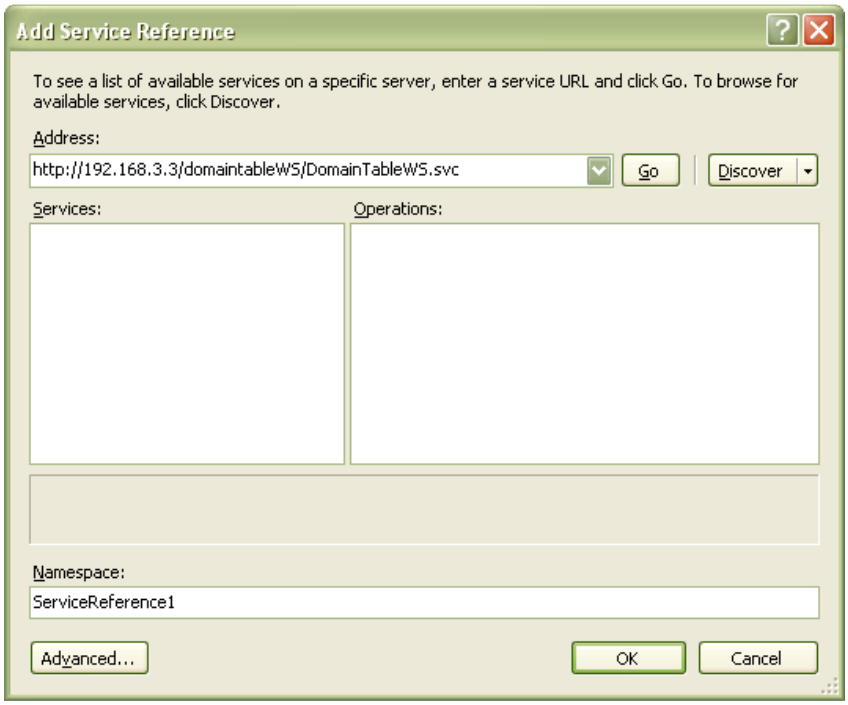
Voeg aan de "References" een "Service Reference" toe (Add Service Reference) en geef de url van de webservice op.

Dit is: **http://domeintabellen-idsw-ws.rws.nl/DomainTableWS.svc**

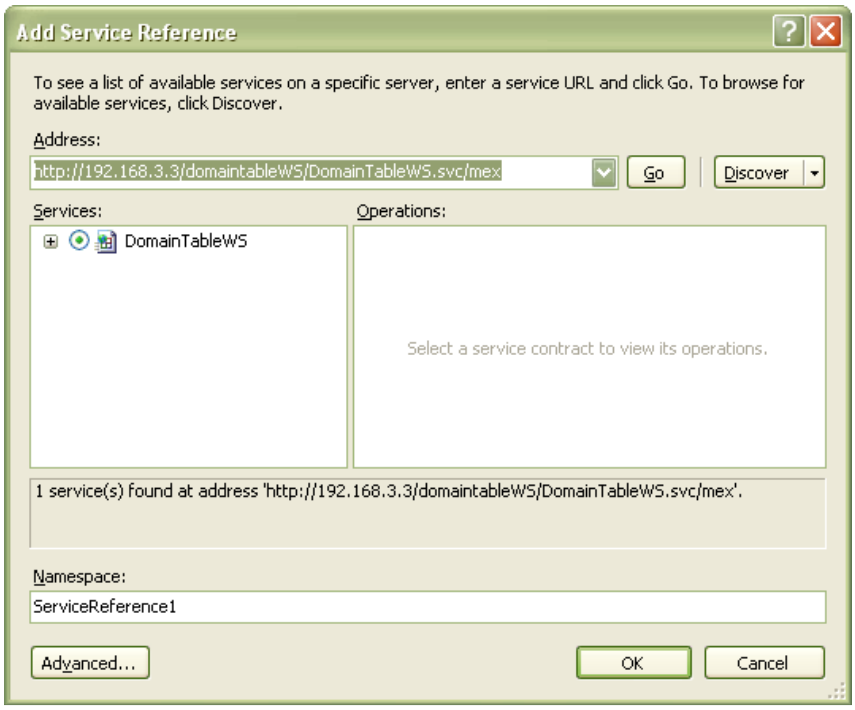
WSDL: http://domeintabellen-idsw-ws.rws.nl/DomainTableWS.svc?wsdl

Let op ! In deze tutorial wordt er nog van uitgegaan dat de webservice benaderbaar is op: http://192.168.3.3/domaintableWS/DomainTableWS.svc

Het volgende scherm verschijnt:

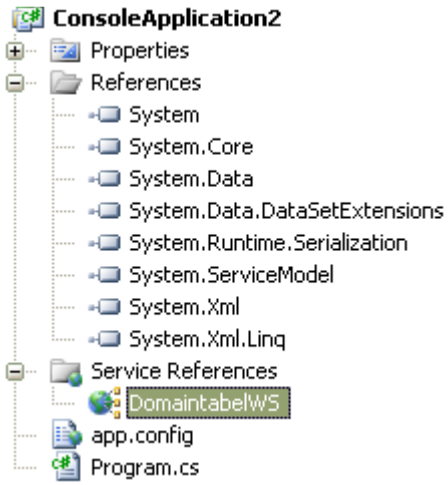


Na het invullen van het "Address" en het drukken op de "Go" knop verschijnt de beschikbare webservice:



Verander eventueel de Namespace in “DomaintabelWS” en druk op “OK”.

De Service reference verschijnt dan in de Solution Explorer. In de client solution zijn de proxy classes gegenereerd en kunnen gebruikt worden binnen de clientapplicatie.



1.3 Controleer gegenereerde app.config

Microsoft Visual Studio genereert door het toevoegen van deze service reference een basis *app.config* toe die nog gewijzigd moet worden:

```
<?xml version="1.0" encoding="utf-8" ?>
  <configuration>
    <system.serviceModel>
      <behaviors>
        <endpointBehaviors>
          <behavior name="basic">
            <dataContractSerializer maxItemsInObjectGraph="1000000" />
          </behavior>
        </endpointBehaviors>
      </behaviors>
      <bindings>
        <basicHttpBinding>
          <binding name="basic" closeTimeout="00:01:00"
openTimeout="00:01:00"
                   receiveTimeout="00:10:00" sendTimeout="00:01:00"
allowCookies="false"
                   bypassProxyOnLocal="false"
hostNameComparisonMode="StrongWildcard"
                   maxBufferSize="200000000" maxBufferPoolSize="524288"
                   maxReceivedMessageSize="200000000"
                   messageEncoding="Mtom" textEncoding="utf-8"
                   transferMode="StreamedResponse"
                   useDefaultWebProxy="true">
            <readerQuotas maxDepth="32" maxStringContentLength="200000000"
                          maxArrayLength="200000000"
                          maxBytesPerRead="4096"
maxNameTableCharCount="16384" />
          </binding>
        </basicHttpBinding>
        ...
      </bindings>
      <client>
        <endpoint
address="http://192.168.3.3/DomainTableWS/DomainTableWS.svc/basic"
                  binding="basicHttpBinding" behaviorConfiguration="basic"
                  bindingConfiguration="basic"
                  contract="DomainTableWSService.DomainTableService"
name="basic" />
        ...
      </client>
    </system.serviceModel>
  </configuration>
```

Controleer en wijzig de met geel gemarkeerde 'endpoint'-adressen, 'bindings' en 'behaviors' in de *app.config*. De noodzakelijke aanpassingen zijn de 'messageEncoding' en 'transferMode' en uiteraard de 'endpoint'. De overige aanpassingen zijn alleen belangrijk indien data wordt opgehaald waarvan de grootte meer is dan standaard bij SOAP webservice is ingesteld.

2. Voorbeelden gebruik methods

In het volgende programma-voorbeeld is het gebruik van de drie mogelijke methods uitgewerkt. Deze methods zijn:

- *GetDomainTableNames*
  Dit is het ophalen van de namen van alle domeintabellen.
- *GetDateLastPublished*
  Dit is het ophalen van de laatste publicatiedatum van een domeintabel. De naam van de domeintabel dient men als parameter mee te geven.
- *GetDomainTable*
  Dit is het ophalen van de administrative informatie en de waarden van een domeintabel. De naam van de domeintabel dient men als parameter mee te geven. Eventueel kan men de peildatum ook opgeven. In dat geval geeft de service informatie van de domeintabel geldig op dat moment.

De volgende C# code toont een heel eenvoudig voorbeeld van het gebruik van de webservice. Foutafhandeling dient in productie software uiteraard toegevoegd te worden.

```
using System;
using ConsoleApplication1.DomaintabelWS; // De gegenereerde proxy classes

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            // Eerst met opgave van het type contract een proxy-object creeren
            var proxy = new DomainTableServiceClient("basic");

            GetDomainTableNames(proxy);
            GetDateLastPublished(proxy, "Parameter");
            GetDomainTable(proxy, "Parameter", null);
        }

        private static void GetDomainTableNames(DomainTableServiceClient proxy)
        {
            // Construeer request (lees alle domeintabelnamen geldig op de
            // gespecificeerde checkdate
            var request = new GetDomainTableNamesRequest
            {
               CheckDate = new DateTime(2010, 5, 22);
            };
            // In het request wordt de parameter “CheckDate” meegegeven
            var response = proxy.GetDomainTableNames(request);

            // Ter controle de namen in het antwoord tonen
            foreach (var name in response.DomainTableNames)
            {
                Console.WriteLine(name);
            }
        }

        private static void GetDateLastPublished(DomainTableServiceClient proxy,
                                                 string domeintable)
        {
            var request = new GetDateLastPublishedRequest
            {
```

```
                DomaintableName = domeintable
            };
            // In het request wordt de parameter "DomaintableName" meegegeven
            var response = proxy.GetDateLastPublished(request);

            // Ter controle het antwoord tonen
            Console.WriteLine("Last published date of domaintable '{0}' is {1}",
                request.DomaintableName,
                response.MostRecentPublicationDate.ToLongDateString());
        }

        private static void GetDomainTable(DomainTableServiceClient proxy,
                                           string domaintable,
                                           DateTime? checkdate)
        {
            var request = new GetDomainTableRequest
            {
                DomaintableName = domaintable,
                CheckDate = checkdate.GetValueOrDefault(DateTime.Now.Date)
            };

            // In het request worden de "DomaintableName" en peildatum
            // meegegeven. Men mag de peildatum weglaten (als null invoeren)
            var response = proxy.GetDomainTable(request);

            // Ter controle het antwoord tonen
            // Eerst de Meta-data van de domeintabel (dwz de structuur)
            Console.WriteLine("Metadata of domaintable {0}:", response.Name);
            foreach (var row in response.DomainTable.MetaData)
            {
                Console.Write("{0}\t{1}\t{2}\t{3}\t{4}", row.ColumnNumber,
                              row.Name, row.FieldLength,
                              row.Required, row.Datatype);
                Console.WriteLine();
            }

            Console.WriteLine("Data of domaintable {0}:", response.Name);

            // Vervolgens de waarderijen van de domeintabel
            foreach (var row in response.DomainTable.Data)
            {
                Console.Write("{0}...{1}: ", row.BeginDate.ToString("dd-MM-yyyy"),
                              row.EndDate.ToString("dd-MM-yyyy"));
                foreach (var field in row.Fields)
                {
                    if (field is StringField)
                        Console.Write("\"{0}\" ", ((StringField)field).Data);
                    else if (field is IntegerField)
                        Console.Write("{0} ", ((IntegerField)field).Data);
                    else if (field is DoubleField)
                        Console.Write("{0} ", ((DoubleField)field).Data);
                    else if (field is BooleanField)
                        Console.Write("{0} ", ((BooleanField)field).Data);
                    else if (field is DateTimeField)
                        Console.Write("{0} ",
                      ((DateTimeField)field).Data.ToString("yyyy-MM-dd HH:mm:ss"));
                }

                Console.WriteLine();
            }
        }
    }
}
```