ICES SOAP data request using matlab

ICES provides a machiene-to-machine (M2M) API for their database with SOAP services, with 2 separate entries, for each of which a WDSL is available that describes in xml-format the available web services requests.

- http://ocean.ices.dk/webservices/hydchem.asmx
- http://ecosystemdata.ices.dk/webservices/EcoSystemWebServices.asmx

SOAP is different than the OGC WxS services and OPeNDAP: SOAP-accepts request are not a simple key-word-pair (KVP) url, but are an xml file that needs to be uploaded to the server. Matlab has the function `createSoapMessage` to construct this xml-message, `callSoapService` to send it to the SOAP server, and `parseSoapResponse` to transforms the xml-answer into a native Matlab data type. The input for `createSoapMessage` is so complex, that Matlab provides a separate funtion `createClassFromWsdl` to create a dedicated SOAP function using the WDSL meta-data on the SOAP services. You have to call `createClassFromWsdl` only *once*, and can then reuse the resulting function for each subsequent SOAP data request. Do not that when the service is updated, you'll need to rerun `createClassFromWsdl` again. Therefore it makes sense also to store the series of `createClassFromWsdl` requests for reuse under version control. `createClassFromWsdl` turns this piece of descriptive xml

```
<wsdl:definitions targetNamespace="http://ocean.ices.dk/webservices/">
<wsdl:documentation>ICES Oceanographic Web Service</wsdl:documentation>
<wsdl:types>
<s:schema elementFormDefault="qualified" targetNamespace="http://ocean.ices.dk/webservices/">
<s:element name="GetICEData">
<s:complexType>
<s:sequence><s:element minOccurs="1" maxOccurs="1" name="ParameterCode" type="tns:ParameterCodeEnum"/>
<s:element minOccurs="1" maxOccurs="1" name="FromYear"      type="s:int"/>
<s:element minOccurs="1" maxOccurs="1" name="ToYear"        type="s:int"/>
<s:element minOccurs="1" maxOccurs="1" name="FromMonth"     type="s:int"/>
<s:element minOccurs="1" maxOccurs="1" name="ToMonth"       type="s:int"/>
<s:element minOccurs="1" maxOccurs="1" name="FromLongitude" type="s:double"/>
<s:element minOccurs="1" maxOccurs="1" name="ToLongitude"   type="s:double"/>
<s:element minOccurs="1" maxOccurs="1" name="FromLatitude"  type="s:double"/>
<s:element minOccurs="1" maxOccurs="1" name="ToLatitude"    type="s:double"/>
<s:element minOccurs="1" maxOccurs="1" name="FromPressure"  type="s:double"/>
<s:element minOccurs="1" maxOccurs="1" name="ToPressure"    type="s:double"/>
</s:sequence>
</s:complexType>
</s:element>
...
</s:schema>
</wsdl:types>
...
<wsdl:service name="ICES_x0020_Oceanographic_x0020_Web_x0020_Service">
</wsdl:service></wsdl:definitions>
```

into this piece of Matlab code

```
function GetICEDataResult = GetICEData(obj,ParameterCode,...
FromYear,ToYear,FromMonth,ToMonth,...
FromLongitude,ToLongitude,FromLatitude,ToLatitude,...
FromPressure,ToPressure)

% Build up the argument lists.
values = {ParameterCode,... % what
          FromYear,ToYear,FromMonth,ToMonth, ... % when
          FromLongitude,ToLongitude,FromLatitude,ToLatitude,... % where
          FromPressure,ToPressure};
names = {'ParameterCode',...
         'FromYear','ToYear','FromMonth', ...
         'ToMonth','FromLongitude','ToLongitude','FromLatitude','ToLatitude',...
         'FromPressure','ToPressure'};
types = { ...
   '{http://ocean.ices.dk/webservices/}ParameterCodeEnum', ...
   '{http://www.w3.org/2001/XMLSchema}int', ...
   '{http://www.w3.org/2001/XMLSchema}int', ...
   '{http://www.w3.org/2001/XMLSchema}int', ...
   '{http://www.w3.org/2001/XMLSchema}int', ...
   '{http://www.w3.org/2001/XMLSchema}double', ...
   '{http://www.w3.org/2001/XMLSchema}double', ...
   '{http://www.w3.org/2001/XMLSchema}double', ...
   '{http://www.w3.org/2001/XMLSchema}double', ...
   '{http://www.w3.org/2001/XMLSchema}double', ...
   '{http://www.w3.org/2001/XMLSchema}double', ...
   };

% Create the message, make the call, and convert the response into a variable.
soapMessage = createSoapMessage( ...
    'http://ocean.ices.dk/webservices/', ...
    'GetICEData', ...
    values,names,types,'document');
soapResponse= callSoapService( ...
    obj.endpoint, ...
    'http://ocean.ices.dk/webservices/GetICEData', ...
    soapMessage);
GetICEDataResult = parseSoapResponse(soapResponse);
```

The Matlab function will create a `soapMessage` request that looks like this

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:soapenc="http://schemas.xmlsoap.org
/soap/encoding/" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <GetICEData xmlns="http://ocean.ices.dk/webservices/">
         <ParameterCode>PSAL</ParameterCode>
         <FromYear      xsi:type="xs:int">2009</FromYear>
         <ToYear        xsi:type="xs:int">2010</ToYear>
         <FromMonth     xsi:type="xs:int">1</FromMonth>
         <ToMonth       xsi:type="xs:int">1</ToMonth>
         <FromLongitude xsi:type="xs:double">-2</FromLongitude>
         <ToLongitude   xsi:type="xs:double">9</ToLongitude>
         <FromLatitude  xsi:type="xs:double">49</FromLatitude>
         <ToLatitude    xsi:type="xs:double">57</ToLatitude>
         <FromPressure  xsi:type="xs:double">0</FromPressure>
         <ToPressure    xsi:type="xs:double">100000</ToPressure>
      </GetICEData>
   </soap:Body>
</soap:Envelope>
```

send it to the ICES server and receive and xml-encoded answer `soapResponse` that will look like this

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001
/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetICEDataResponse xmlns="http://ocean.ices.dk/webservices/">
<GetICEDataResult>
<ICEData>
        <Longitude>-0.81470000743865967</Longitude>
        <Latitude>49.693500518798828</Latitude>
        <DateTime>2009-01-16T22:48:00</DateTime>
        <Pressure>1</Pressure>
        <Value>34.94</Value>
</ICEData>
<ICEData>
        <Longitude>-0.81470000743865967</Longitude>
        <Latitude>49.693500518798828</Latitude>
        <DateTime>2009-01-16T22:48:00</DateTime>
        <Pressure>10.9</Pressure><Value>34.97</Value>
</ICEData>
...
</GetICEDataResult>
</GetICEDataResponse>
</soap:Body></soap:Envelope>
```

In OpenEarthTools we made an ICES Matlab toolbox to do this for you under the hood. It is free, but you need to register first to get it. The first function `createicesservices` is 'hidden' from the contents.m, because you do not need it, we already ran it once and provide the resulting SOAP message constructors. We made a set of wrappers for each ICES function, that calls the trio `createSoapMessage`, `callSoapService` and `parseSoapResponse`. The result will be ready-to-use Matlab structs. As a check, it optionally plots the data in kml for ready viewing in Google Earth.

```
   [D,A] = getICESdata('ParameterCode','PSAL','t0',datenum(2009,1,1),'t1',datenum(2010,1,1),...
               'lon',[-2  9],... % longitude bounding box
               'lat',[49 57],... % latitude  bounding box
               'p'  ,[0 1e5],'kml','salinity.kml')
```

This will result in Matlab array of struct (tuples) as popular with database-minded users

```
D =
5225x1 struct array with fields:
    Longitude
    Latitude
    DateTime
    Pressure
    Value
    datenum
    name
    units
```

but optionally also in the computationally more efficient struct of arrays, that can readily be plotted, as popular with Matlab-minded users

```
A =
         code: 'PSAL'
         name: 'Salinity'
        units: 'psu'
    Longitude: [1x5225 double]
     Latitude: [1x5225 double]
     Pressure: [1x5225 double]
        Value: [1x5225 double]
      datenum: [1x5225 double]
scatter(A.Longitude,A.Latitude,20,A.Value)
title ([A.name,'[',A.units,']'])
axislat(A.Longitude(1))
tickmap('ll')
```

This code takes about 75 s to run to get order 5000 data points, estimated by Matlabs `profile`.

SOAP function	`profile` time
`createSoapMessage`	0.03 s
`callSoapService`	0.8 s
`parseSoapResponse`	70 s

This timing example illustrates that xml is not a very efficient method of transporting large data over the web. The data takes less then a second to go from the from the ICES server to the client (you, behind Matlab) but then Matlab needs over 1 minute to turn the xml stream `soapResponse` into numeric matrices. SOAP was designed for processes were small amounts of data need to be transported safely over the web e.g. (purchasing something with a credit card, booking a train or plane ticket), but it performs not so well when large amounts of data (or even 'relatively' largish, only 5000 in this case) that do not even require much in term of safety (this ICES data is open anyway). The biggest time consumer of parsing the xml into data is actually the line of Matlab code that searches for ":" to check for xml-namespaces. The xml parser of the Python equivalent is twice as fast, which leads to double performance due to the dominant role of xml parsing performance in the overall timing.

See also: ICES SOAP data request using Python,NOAA CO-OPS SOAP services as implemented in DelftDashBoard