

Copyright 2006 Krengel Technologies Inc.

Author: Aaron Bartell

Program: EXAMPLE,GETURI2

```

//*****
// @Author: Aaron Bartell
// @Creation Date:
// @Descr: Compose a SOAP xml document to send to a web service hosted on www.w3schools.com.
//          Use RXS_getUri to send the document to the web service making sure to specify
//          all of the appropriate headers. And lastly, receive the response back from the
//          web service and parse the document to obtain the end result data.
//
//          The web service is simple in concept in that it is passing in a Fahrenheit
//          temperature and converting it to Celsius and returning that value to this program.
// @Notes:
//*****
H dftactgrp(*no) bnddir('RXSBND')

/copy rxs,RXSCp

//-----
// Local Prototypes
//-----
D compose          pr
D transmit         pr
D parse           pr

D allHandler      pr
D pType           value like(RXS_Type)
D pXPath          value like(RXS_XPath)
D pData           value like(RXS_XmlData)
D pDataLen        value like(RXS_Length)

D errHandler      pr
D pCurLine       10i 0 value
D pCurCol        10i 0 value
D pErrStr         1024a value varying

//-----
// Global Variables
//-----
D gError          ds          likeds(RXS_Error)
D gReqFile        s          256a varying inz('geturi2_request.xml')
D gRspFile        s          256a varying inz('geturi2_response.xml')
D gCelsiusValue   s          10i 0

//-----
// Mainline: First initialize all necessary fields.
//
//          Call compose() to compose the SOAP xml document that is to be sent to the
//          www.w3schools.com web service using the RXS Template Engine.
//
//          Call transmit() to send the SOAP XML file that was created to www.w3schools.com
//
//          Call parse() to parse the xml response to retrieve the response data.
//
//          Send a message to the job log using RXS_log stating the results. Do a DSPJOBLOG
//          to view the message.
//-----
/free

clear gError;

compose();
if gError.code <> *blanks;
return;
endif;

transmit();
if gError.code <> *blanks;
return;
endif;

parse();

RXS_log(RXS_DIAG: 'Result:' + %char(gCelsiusValue));

*inlr = *on;

/end-free
```

```

//-----
// @Author: Aaron Bartell
// @Created: 2005-08-03
// @Desc: Compose the xml necessary to send to the www.w3schools.com web service. Template
// geturi2.tpl can be found in /www/myrxs/templates/geturi2.tpl.
//
// RXS_initTplEng is initializing the template engine and allows you to specify where
// the composed document is located. In this case we want to put the composed document
// in the IFS (i.e. RXS_FILE), and we state the name of the file by using variable
// gReqFile which has the value of 'geturi2_request.xml'. Note that this will place it
// in the default transaction directory of /www/myrxs/trans/geturi2_request.xml.
// The first *omit parm can be replaced with a code page you want the IFS file to
// be created as. The default will be fine in most cases. The second and third *omit
// parms allow you to temporarily specify different template and transaction
// directories. For instance maybe you want to put your templates in
// /mycompany/templates/ and put transactions in /mycompany/transactions/. The last
// parameter is telling the template engine to be in debug mode which will place
// helpful information into the job log for debugging purposes.
//
// RXS_loadTpl is bringing an IFS template file into the Template Engine to be used
// later on with calls to RXS_updvar and RXS_wrtSection. Note that the relative
// location of geturi2.tpl is used instead of a full qualified path because geturi2.tpl
// is located in the default template directory of /www/myrxs/templates/.
//
// RXS_updvar is replacing the variable place holder in the template
// (i.e. /%fahrenheit%/) with %char(100). Note that you don't have to specify the
// beginning and ending variable delimiters in your RPG program (i.e. /% and %/).
//
// RXS_wrtSection is taking the section specified and writing it out to the IFS file.
// In this case there is only one section being written out, 'CONTENT'. The second
// parameter is telling the Template Engine to flush the buffers contents and send it
// to the IFS file.
//
// The on-error clause is monitoring for any errors that occur and will place the error
// in variable gError.
//-----

```

```

P compose          b
D  compose          pi
/free

```

```
monitor;
```

```

RXS_initTplEng(RXS_STMF: gReqFile: *omit: *omit: *omit: *on);
RXS_loadTpl('geturi2.tpl');

```

```

RXS_updVar('fahrenheit': %char(100));
RXS_wrtSection('CONTENT': *on);

```

```

on-error;
gError = RXS_catchError();
endmon;

```

```
/end-free
```

```
P          e
```

```

-----
// @Author: Aaron Bartell
// @Created: 2005-08-03
// @Desc: This sub procedure will take the SOAP XML request composed with the Template Engine
// in sub proc compose() and transmit it to the end point (say web service).
// The API to be used here is RXS_getUri. Before RXS_getUri can be called it must
// be primed with the information necessary to invoke the web service.
//
// inParms.URI is the qualified http location of the web service to be invoked.
//
// inParms.ReqType is stating the input will be read from a stream file.
//
// inParms.RspType is stating the output will be sent to a stream file.
//
// inParms.ReqStmf is specifying the location of the stream file to send on the
// request. Note that if the request document resides in the default transaction
// directory then a relative path can be used.
//
// inParms.RspStmf is specifying the location of the stream file to receive the output
// or response of the web service request. Note that if the response document is to
// reside in the default transaction directory then no path prefix is necessary.
//
// inParms.ContentType is a http header that will be sent telling the web server on the
// other end what type of content is contained within this request. In this case we
// are sending 'text/xml'. This will be the most common ContentType used for web
// services.
//
// inParms.SprHead is stating that we want the http headers that are returned in the
// response to be separated from the gRspFile. We want to do this because the xml
// parser that is going to process the gRspFile will throw an error if the http
// headers are left in gRspFile as those headers are not valid xml.
//
// inParms.ReqMeth is stating that POST will be used for this request (vs. GET or HEAD)
// POST will be the most common method for calling web services.
//
// inParms.Nbr is stating the number of additional http headers that need to be sent
// to the web service. In this case only one needs to be sent - 'SOAPAction'.
//
// inParms.UsrHdr is the name of the http header that is to be sent - 'SOAPAction'.
//
// inParms.UsrHdrDta is the value of the http header that is to be sent -
// "http://tempuri.org/FahrenheitToCelsius".
//
// RXS_getUri is invoking the web service with the data contained within inParms.
// outDta will not be used in this case as we are sending the response to a stream
// file in the IFS. outHead will contain the http headers sent back from the web
// services server.
-----

```

```

P transmit      b
D transmit      pi

D inParms       ds          likeds(RXS_GetUriIn) inz
D outDta        s          like(RXS_GetUriOut)
D outHead       s          like(RXS_GetUriHead)
/free

```

```

monitor;
  inParms.URI = 'http://www.w3schools.com/webservices/tempconvert.asmx';
  inParms.ReqType = RXS_STMF;
  inParms.RspType = RXS_STMF;
  inParms.ReqStmf = gReqFile;
  inParms.RspStmf = gRspFile;
  inParms.ContType = 'text/xml';
  inParms.SprHead = RXS_YES;

  inParms.ReqMeth = RXS_POST;

  inParms.NbrHdrs = 1;
  inParms.UsrHdr(1) = 'SOAPAction';
  inParms.UsrHdrDta(1) = '"http://tempuri.org/FahrenheitToCelsius"';

```

```

  RXS_getUri(inParms: outDta: outHead);
on-error;
  gError.code = '100';
  gError.text = 'Error occurred during the parsing of the document.';
endmon;

```

```

/end-free

```

```

P          e

```

```

-----
// @Author: Aaron Bartell
// @Created: 2005-08-03
// @Desc: This sub procedure is used to prep and invoke the parser with the appropriate
// information.
//
// RXS_allElemContentHandler is telling the parser your program wants to be notified of
// all parsing events involving element content. A procedure pointer is required that
// tells the parser which local sub procedure to call when it finds element content.
//
// RXS_parse is telling the parser to start parsing the document. The first parm is
// specifying the file in the IFS to be parsed and the second parm is stating the
// value in the first parm is indeed a file. RXS_parse also has the ability to parse
// an RPG variable containing XML so that is why the second parm is needed to declare
// what is contained in the first parm. The third parm is telling the parser what
// local sub procedure to call in the event it encounters invalid xml.
-----

```

```

P parse          b
D parse          pi
/free

```

```

monitor;
  RXS_allElemContentHandler( %paddr(allHandler) );
  RXS_parse(gRspFile: RXS_STMF: %paddr(errHandler));
on-error;
  gError.code = '100';
  gError.text = 'Error occurred during the parsing of the document.';
endmon;

```

```

/end-free
P          e

```

```

-----
// @Author: Aaron Bartell
// @Created: 2005-08-03
// @Desc: This is the local sub procedure specified on RXS_allElemContentHandler. This sub
// procedure will get called by the parser each time element content is encountered.
// Each time this sub procedure is called there will be information passed so we know
// exactly what the parser found and where it found it.
//
// For instance, the pType parm will tell us what type of event is occurring (i.e.
// RXS_ELEMBEGIN, RXS_ATTR, RXS_ELEMCONTENT, or RXS_ELEMEND). In this specific scenario
// it will only ever be RXS_ELEMCONTENT because that is the only thing we told the
// parser we wanted to be notified of.
// The pXPath parm will tell us the full path of the element content it found. As you
// can see in the 'if' statement we are checking the pXPath variable to see if we are
// currently processing the <FahrenheitToCelsiusResult> element.
// If this is the <FahrenheitToCelsiusResult> element then we want to obtain the data
// and put it in a global variable. RXS_charToNbr is a simple sub procedure to ease the
// conversion of character to numeric.
// pData contains the actual data that was parsed which in this case will be '38'.
// pDataLen contains the data length which in this case will be 2.
-----

```

```

P allHandler     b
D allHandler     pi
D pType          value like(RXS_Type)
D pXPath         value like(RXS_XPath)
D pData          value like(RXS_XmlData)
D pDataLen       value like(RXS_Length)
/free

```

```

// This 'if' statement is checking to see if the pXPath variable contains the element we
// are looking for. The RXS_ELEMCONTENT contains a value of '/' which signifies we are
// looking for the element content of <FahrenheitToCelsiusResult>. If we were looking to
// be notified of the beginning of <FahrenheitToCelsiusResult> we would specify
// RXS_ELEMBEGIN, and in the same manner if we wanted to be notified of the ending
// </FahrenheitToCelsiusResult> we would specify RXS_ELEMEND.

```

```

if pXPath =
  '/soap:Envelope/soap:Body/FahrenheitToCelsiusResponse' +
  '/FahrenheitToCelsiusResult' + RXS_ELEMCONTENT;

  gCelsiusValue = RXS_charToNbr(pData);
endif;

```

```

/end-free
P          e

```

```

-----
// @Author: Aaron Bartell
// @Created: 2005-08-03
// @Desc: If an error occurs during the parsing of xml then this sub procedure will be
// called by the xml parser. An example of an error would be un-matched begin and end
// tags. In this case the gError data structure variable is being occupied with the
// appropriate info so the mainline of this program knows that normal processing
// should not continue.
-----

```

```

P errHandler      B
D errHandler      PI
D pCurLine       10i 0 value
D pCurCol        10i 0 value
D pErrStr         1024a value varying
/free

```

```

gError.code = 'GETURI2001';
gError.severity = 100;
gError.pgm = 'GETURI2.errHandler';
gError.text =
  'Line:' + %char(pCurLine) +
  ' Column:' + %char(pCurCol) +
  ' ' + pErrStr;

```

```

/end-free
P      E

```

Contents of template geturi2.tpl

```

/$CONTENT
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <FahrenheitToCelsius xmlns="http://tempuri.org/">
      <Fahrenheit>/%fahrenheit%</Fahrenheit>
    </FahrenheitToCelsius>
  </soap:Body>
</soap:Envelope>

```

Contents of transaction document geturi2_request.xml

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <FahrenheitToCelsius xmlns="http://tempuri.org/">
      <Fahrenheit>100</Fahrenheit>
    </FahrenheitToCelsius>
  </soap:Body>
</soap:Envelope>

```

Contents of transaction document geturi2_response.xml

```

<?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>
    <FahrenheitToCelsiusResponse xmlns="http://tempuri.org/">
      <FahrenheitToCelsiusResult>38</FahrenheitToCelsiusResult>
    </FahrenheitToCelsiusResponse>
  </soap:Body>
</soap:Envelope>

```