SOAPLogicsheet

The SOAP logicsheet is able to create a SOAP call, then execute it, and include the results in place of the call.

Compare this to the ESQL logicsheet: It creates a DB call, executes it, and includes the results in place of the call.

What is SOAP

SOAP is a protocol definition, allowing to call "services" over a HTTP connection.

The SOAP request is a XML message, the SOAP reply too.

The XML part of a typical SOAP request looks like:

```
<?xml version="1.0" encoding="UTF-8?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
  <getPartnerData>
   <getPartnerDataRequest>
    <partnerNo>1</partnerNo>
   </getPartnerDataRequest>
  </getPartnerData>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

This XML data must be POSTed to the Webservice's URL
(e.g. "http://webservice.somewhere.org/partner"). In the HTTP request there has to be a HTTP header "SOAPAction".

After serving the SOAP request the caller will get a HTTP reply which could look like:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
 <soapenv:Body>
  <getPartnerDataResponse
        soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" >
   <getPartnerDataReturn>
    <bpPerson>
     <birthdate>1960-01-01</birthdate>
     <nationality>DE</nationality>
     <lastname>Muster</lastname>
     <initials>E.E.</initials>
     <sex>1</sex>
     <fullname>Dr. Max E.E. Muster</fullname>
     <firstname>Max</firstname>
     <partnerNo>1</partnerNo>
    </bpPerson>
   </getPartnerDataReturn>
  </getPartnerDataResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

What is the SOAP logicsheet

The SOAP logicsheet is a feature of XSP, which allows to include SOAP calls in a xsp:page.

The SOAP call is executed and replaced by the SOAP results.

Everything written here has been tested with Cocoon 2.0.4.

The SOAP call from above could be included in a xsp:page as:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsp:page language="java"
  xmlns:xsp="http://apache.org/xsp"
  xmlns:xsp-request="http://apache.org/xsp/request/2.0"
  xmlns:soap="http://apache.org/xsp/soap/3.0"
  xmlns:xscript="http://apache.org/xsp/xscript/1.0"
  xmlns:log="http://apache.org/xsp/log/2.0">
 <page>
  <soap:call url="http://webservices.somewhere.org:80/partner">
   <getPartnerData>
    <getPartnerDataRequest>
     <partnerNo><xsp-request:get-parameter name="partnerID"/></partnerNo>
    </getPartnerDataRequest>
   </getPartnerData>
  </soap:call>
 </page>
</xsp:page>
```

Here, the input parameter for the "query" (partnerNo) comes from the request parameter PartnerID.

You may also need to provide a method using the *method* attribute of the soap call. This will become the SOAPAction http-header.

After execution of the <map:generate type="serverpages"> of the pipeline,

the <soap:call> will be replaced by the SOAP result, and the next step of pipeline processing

will see the following:

```
<?xml version="1.0" encoding="UTF-8" ?>

<page xmlns:xsp="http://apache.org/xsp"
         xmlns:xscript="http://apache.org/xsp/xscript/1.0"
         xmlns:soap="http://apache.org/xsp/soap/3.0"
         xmlns:xsp-session="http://apache.org/xsp/session/2.0"
         xmlns:xsp-request="http://apache.org/xsp/request/2.0"
         xmlns:log="http://apache.org/xsp/log/2.0">
 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <soapenv:Body>
   <getPartnerDataResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" >
    <getPartnerDataReturn>
     <bpPerson>
      <birthdate>1960-01-01</birthdate>
      <nationality>DE</nationality>
      <lastname>Muster</lastname>
      <initials>E.E.</initials>
      <sex>1</sex>
      <fullname>Dr. Max E.E. Muster</fullname>
      <firstname>Max</firstname>
      <partnerNo>4711</partnerNo>
     </bpPerson>
    </getPartnerDataReturn>
   </getPartnerDataResponse>
  </soapenv:Body>
 </soapenv:Envelope>
</page>
```

How to connect to a web service via https://

The current implementation of the SOAP logicsheet is not able to access webservices trough secure socket connections (https://... does not work!) But it is not neccessary to extend Cocoon, you can use stunnel to enable secure connections.

The SOAP logic sheet calls a web service on a local http port, where stunnel is listening in client mode. This program does the ssl encryption and communicates wich the real web service on the remote server through https://

Steps to do:

1. get stunnel from stunnel.org

2. edit stunnel.conf

```
foreground=yes
client=yes
debug=3
pid=/tmp/stunnel.pid
[https]
accept=localhost:8585
connect=somewhere.secure.org:443
transparent=no
```

In this example web service calls to http://localhost:8585 are redirected to https://somewhere.secure.org

3. start stunnel (stunnel stunnel.conf)

Thanks to Larissa for this information

ChristophOberle

Q: How do you make a decision about what to do AFTER you've made the SOAP request? Ie, depending upon what the soap server returns, you want the freedom to decide on what to do next in cocoon.
All examples I've seen (eg the google, amazon examples) don't cater for handling errors, they just assume success and then format into HTML using an XSL stylesheet. In terms of presentation, I can understand this (showing success or failure), but I want to be able to control what happens in terms of more actions to run, or redirecting, or whatever.

I was wondering the same, and have been playing with ways of interfacing with web services from flow script. Take a look at the FlowAndWebServices I am drafting on the subject, all comments welcome. LukeHubbard