Exploring Web Services

Tripod workshop on Perl, SQL and Web Services

November 16-20, 2009

Peter Fischer Hallin, November 19, 2009
• Peter Fischer Hallin, Research Scientist, Novozymes A/S - PhD from CBS, December 2009 :-)

• Comparative genomics of in-house genome projects as well as mining public genome data.

• Novozymes is the world’s largest manufacturer of industrial enzymes, with a global market share of 47% (2008)

• Novozymes employs some 5,000 people half of which are based in Denmark.

• 700 products in 130 countries, covering industrial enzymes for use in washing detergents, biofuel, brewing, baking, detergent, animal feed
Today’s challenges

- Increasing **size of public sequence databases** due to faster and cheaper sequencing methods

- Increasing **number of bioinformatic prediction methods**.

- Increasing number of **biotechnological data types**: Protein-protein interaction, transcriptomics data, proteomics data, disease genes/SNP information, cell-cycle regulation data

- More sequence data + more tools + new experimental techniques $\rightarrow$ **new integration solutions are required**

- To exploit the massive amounts of data, **interoperability and connectivity** are central issues. EMBRACE has provided better understanding of the needs of the community and awareness of Web Services technology.
Web applications

- HTML based software designed to interact with the user - in most cases this involves human interpretation and navigation.
- In most cases, user-friendly
- In most cases, do not require special skills to operate
- **In most cases, web pages do not make sense to computers.** Even more true for modern web design using Ajax and advanced Java scripting and Java applets.
Web Services

- Are software designed to enable computer-to-computer interaction
- Should aim to enhance interoperability between different systems
- Exchange objects which are well defined
- Consist of methods / operations that are well defined
- Often exchanging data using SOAP over HTTP.
SOAP messages

**request**

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
     <getProductDetails xmlns="http://warehouse.example.com/ws">
       <productID>827635</productID>
     </getProductDetails>
   </soap:Body>
</soap:Envelope>
```

**response**

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
     <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
       <getProductDetailsResult>
         <productName>Toptimate 3-Piece Set</productName>
         <productID>827635</productID>
         <description>3-Piece luggage set. Black Polyester.</description>
         <price currency="NIS">96.50</price>
         <inStock>true</inStock>
       </getProductDetailsResult>
     </getProductDetailsResponse>
   </soap:Body>
</soap:Envelope>
```
Some key concepts

- **XML** (Extensible Markup Language): Format for input and output, XSD, and WSDL.
- **Request/Response**: XML messages generated by the client / server.
- **SOAP** (originally Simple Object Access Protocol) The is the protocol defining the structure of the message (header, body, name spaces etc).
- **XSD** (XML Schema Definition): Defines the exact object/data type within the message.
- **WSDL** (Web Services Description Language) file: Most often published by the service provider. Strictly speaking, this is not required for SOAP communication, but almost always used to provide the client with understanding of the resource. The WSDL is also an XML document.
The WSDL - an XML explaining the resource

A single document explaining all what the client needs to know about the service, in order to communicate with it.

- Declares the name space of the resource avoiding clashes
- Defines all the operations that the service can perform
- Defines the request and response messages of each operation.
- Defines where messages should be sent (endpoint)
- Contains documentation of elements and attributes

WSDLs can be very complex XML structures. Most IT specialists choose to use specialized software to author WSDL (e.g. XML Spy) or have them machine-generated
The WSDL - explaining the resource

- `<wsdl:definitions>`
  - `<wsdl:types>`
    - `<schema>`
      - declaration of elements and attributes
  - `<message>`
  - `<portType>`
  - `<binding>`
  - `<service>`
    - `<endpoint>`

External XSD files

XML
The XSD - declaring complex data types

The XSD section contains the declaration of the sometimes complex XML structures within the message. Not only the structure of elements and attributes but also facets, restrictions and even regular expressions of content is allows.

The XSD is where your SOAP client (e.g. Perl, C, Python, Java) learns how the data structures are composed.
<?xml version="1.0"?>
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:tns="http://www.cbs.dtu.dk/ws/WSGenomeAtlas_3_0_ws2"
    xmlns:genomeatlas="http://www.cbs.dtu.dk/ws/ws-genomeatlas"
    xmlns:common="http://www.cbs.dtu.dk/ws/ws-common"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://www.cbs.dtu.dk/ws/WSGenomeAtlas_3_0_ws2">
  <xsd:schema>
    <xsd:element name="getSeq" type="genomeatlas:getSeq"/>
    <xsd:element name="getSeqResponse" type="genomeatlas:getSeqResponse"/>
  </xsd:schema>
</wsdl:definitions>
Granularity in data types

<sequences>
  <seq>
    <id>P09493-2|TPM1_HUMAN Isoform 2 of P09493 - Homo sapiens (Human)</id>
    <sequence>MDAIKKKMQMLKDKNALDRAEQAEADKKAAEDRSQLEDELVSLQKKLGTDLEDKLYSEALKDAQEKLLEAEKKATDAEADV</sequence>
  </seq>
  <seg>
    <id>sp_vs|P09493-2|TPM1_HUMAN Isoform 2 of P09493 - Homo sapiens (Human)</id>
    <sequence>MCRLRIFLRTASSEHLHERKLRTAEADVASELNRRILVEELDRAQERLATALQKLEAEKKAEDESERGMKVIERSRAQKDEE</sequence>
  </seg>
  <seg>
    <id>sp_vs|P09493-3|TPM1_HUMAN Isoform 3 of P09493 - Homo sapiens (Human)</id>
    <sequence>MDAIKKKMQMLKDKNALDRAEQAEADKKAAEDRSQLEDELVSLQKKLGTDLEDKLYSEALKDAQEKLLEAEKKATDAEAFP</sequence>
  </seg>
  <seg>
    <id>sp_vs|P09493-4|TPM1_HUMAN Isoform 4 of P09493 - Homo sapiens (Human)</id>
    <sequence>MDAIKKKMQMLKDKNALDRAEQAEADKKAAEDRSQLEDELVSLQKKLGTDLEDKLYSEALKDAQEKLLEAEKKATDAEAFP</sequence>
  </seg>
  <seg>
    <id>sp_vs|P09493-5|TPM1_HUMAN Isoform 5 of P09493 - Homo sapiens (Human)</id>
    <sequence>MAG3SSEAVRRKIRSLQEQADAAEBRAGTLQRELHERKLRTAEADVASELNRRILVEELDRAQERLATALQKLEAAERF</sequence>
  </seg>
</sequences>

<sequences>
  <seq>
    <id>P09493-2|TPM1_HUMAN Isoform 2 of P09493 - Homo sapiens (Human)</id>
    <sequence>MDAIKKKMQMLKDKNALDRAEQAEADKKAAEDRSQLEDELVSLQKKLGTDLEDKLYSEALKDAQEKLLEAEKKATDAEADV</sequence>
  </seq>
  <seg>
    <id>sp_vs|P09493-2|TPM1_HUMAN Isoform 2 of P09493 - Homo sapiens (Human)</id>
    <sequence>MCRLRIFLRTASSEHLHERKLRTAEADVASELNRRILVEELDRAQERLATALQKLEAEKKAEDESERGMKVIERSRAQKDEE</sequence>
  </seg>
  <seg>
    <id>sp_vs|P09493-3|TPM1_HUMAN Isoform 3 of P09493 - Homo sapiens (Human)</id>
    <sequence>MDAIKKKMQMLKDKNALDRAEQAEADKKAAEDRSQLEDELVSLQKKLGTDLEDKLYSEALKDAQEKLLEAEKKATDAEAFP</sequence>
  </seg>
  <seg>
    <id>sp_vs|P09493-4|TPM1_HUMAN Isoform 4 of P09493 - Homo sapiens (Human)</id>
    <sequence>MDAIKKKMQMLKDKNALDRAEQAEADKKAAEDRSQLEDELVSLQKKLGTDLEDKLYSEALKDAQEKLLEAEKKATDAEAFP</sequence>
  </seg>
  <seg>
    <id>sp_vs|P09493-5|TPM1_HUMAN Isoform 5 of P09493 - Homo sapiens (Human)</id>
    <sequence>MAG3SSEAVRRKIRSLQEQADAAEBRAGTLQRELHERKLRTAEADVASELNRRILVEELDRAQERLATALQKLEAAERF</sequence>
  </seg>
</sequences>

<sequences>
  <seq>
    <id>P09493-2|TPM1_HUMAN Isoform 2 of P09493 - Homo sapiens (Human)</id>
    <sequence>MDAIKKKMQMLKDKNALDRAEQAEADKKAAEDRSQLEDELVSLQKKLGTDLEDKLYSEALKDAQEKLLEAEKKATDAEADV</sequence>
  </seq>
  <seg>
    <id>sp_vs|P09493-2|TPM1_HUMAN Isoform 2 of P09493 - Homo sapiens (Human)</id>
    <sequence>MCRLRIFLRTASSEHLHERKLRTAEADVASELNRRILVEELDRAQERLATALQKLEAEKKAEDESERGMKVIERSRAQKDEE</sequence>
  </seg>
  <seg>
    <id>sp_vs|P09493-3|TPM1_HUMAN Isoform 3 of P09493 - Homo sapiens (Human)</id>
    <sequence>MDAIKKKMQMLKDKNALDRAEQAEADKKAAEDRSQLEDELVSLQKKLGTDLEDKLYSEALKDAQEKLLEAEKKATDAEAFP</sequence>
  </seg>
  <seg>
    <id>sp_vs|P09493-4|TPM1_HUMAN Isoform 4 of P09493 - Homo sapiens (Human)</id>
    <sequence>MDAIKKKMQMLKDKNALDRAEQAEADKKAAEDRSQLEDELVSLQKKLGTDLEDKLYSEALKDAQEKLLEAEKKATDAEAFP</sequence>
  </seg>
  <seg>
    <id>sp_vs|P09493-5|TPM1_HUMAN Isoform 5 of P09493 - Homo sapiens (Human)</id>
    <sequence>MAG3SSEAVRRKIRSLQEQADAAEBRAGTLQRELHERKLRTAEADVASELNRRILVEELDRAQERLATALQKLEAAERF</sequence>
  </seg>
</sequences>
Granularity in data types
Flattening the learning curve ...
Exercise workflow

- Download genome sequence (getSeq)
- Predict protein coding genes (runProdigal)
- Poll queue until job has finished (pollQueue)
- Obtain results (prodigalFetchResult)
SoapUI demo ...
Inspecting a resource using SoapUI
The programmatic way
#!/usr/bin/perl
use Dumper;
use strict;
use XML::Compile;
use XML::Compile::WSDL11;
use XML::Compile::Transport::SOAPHTTP;

# parse the XML of the WSDL file
warn "parsing WSDL\n";
my $genomeAtlasProxy = XML::Compile::WSDL11->new($genomeAtlasWSDL);

<?xml version="1.0"?>
<wSDL:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.cbs.dtu.dk/ws/WSGenomeAtlas_3_0_ws2"
xmlns:genometlas="http://www.cbs.dtu.dk/ws/ws-genomeatlas"
xmlns:common="http://www.cbs.dtu.dk/ws/ws-common"
xmlns="http://schemas.xmlsoap.org/wsdl/
targetNamespace="http://www.cbs.dtu.dk/ws/WSGenomeAtlas_3_0_ws2">
<wSDL:types>
  <xsd:schema>
    <xsd:import namespace="http://www.cbs.dtu.dk/ws/ws-genomeatlas"
      schemaLocation="ws_genomeatlas_3_0_ws2.xsd"/>
    <xsd:import namespace="http://www.cbs.dtu.dk/ws/ws-common"
      schemaLocation="../common/ws_common_1_0b.xsd"/>
  </xsd:schema>
</wSDL:types>
Appending schemas to WSDL proxy

Perl

```perl
warn "parsing XSD\n";
my $genomeAtlasXSD = XML::LibXML->new->parse_file('http://www.cbs.dtu.dk/ws/GenomeAtlas/ws_genomeatlas_3_0_ws2.xsd');
my $commonXSD = XML::LibXML->new->parse_file('http://www.cbs.dtu.dk/ws/common/ws_common_1_0b.xsd');

# append the XSD's to the proxy
warn "appending XSD\n";
$genomeAtlasProxy->schemas->importDefinitions ($genomeAtlasXSD);
$genomeAtlasProxy->schemas->importDefinitions ($commonXSD);

<?xml version="1.0"?>
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://www.cbs.dtu.dk/ws/WSGenomeAtlas_3_0_ws2"
 xmlns:genomeatlas="http://www.cbs.dtu.dk/ws/ws-genomeatlas"
 xmlns:common="http://www.cbs.dtu.dk/ws/ws-common"
 xmlns="http://schemas.xmlsoap.org/wsd1/
 targetNamespace="http://www.cbs.dtu.dk/ws/WSGenomeAtlas_3_0_ws2">
 <wsdl:types>
  <xsd:schema>
   <xsd:import namespace="http://www.cbs.dtu.dk/ws/ws-genomeatlas"
 schemaLocation="ws_genomeatlas_3_0_ws2.xsd"/>

   <xsd:import namespace="http://www.cbs.dtu.dk/ws/ws-common"
 schemaLocation="/common/ws_common_1_0b.xsd"/>
  </xsd:schema>

  <schema
 targetNamespace="http://www.cbs.dtu.dk/ws/WSGenomeAtlas_3_0_ws2"
 xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsd1/"
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/encoding/">

   <xsd:element name="getSeq" type="genomeatlas:getSeq"/>
   <xsd:element name="getSeqResponse" type="genomeatlas:getSeqResponse"/>
  </schema>
 </wsdl:types>
```
Compiling operations

Perl

```perl
# compile the operations
warn "compiling operations\n";
my $getSeq = $genomeAtlasProxy->compileClient('getSeq');
```

WSDL

```xml
<message name="getSeqRequest">
    <part name="parameters" element="tns:getSeq"/>
</message>
<message name="getSeqResponse">
    <part name="parameters" element="tns:getSeqResponse"/>
</message>

<portType name="WSGenomeAtlas_3_0_ws2">
    <operation name="getSeq">
        <input name="getSeqRequest" message="tns:getSeqRequest"/>
        <output name="getSeqResponse" message="tns:getSeqResponse"/>
    </operation>
</portType>
```
Executing operation - actions and endpoint

Perl

```perl
# fetch the sequence of the plasmid
warn "fetching sequence data\n";
my $getSeqResponse = $getSeq->(
    parameters => {
        parameters => {
            genbank => 'AB011549'
        }
    }
);
```

WSDL

```xml
<binding name="WSGenomeAtlas_3_0_ws2" type="tns:WSGenomeAtlas_3_0_ws2">
    <operation name="getSeq"><soapAction="http://www.cbs.dtu.dk/ws/WSGenomeAtlas_3_0_ws2#getSeq"/>
        <input>
            <soapenv:body use="literal"/>
        </input>
        <output>
            <soapenv:body use="literal"/>
        </output>
    </operation>
</binding>

<service name="WSGenomeAtlas_3_0_ws2">
    <port name="WSGenomeAtlas_3_0_ws2" binding="tns:WSGenomeAtlas_3_0_ws2">
        <soapenv:address location="http://ws.cbs.dtu.dk:80/cgi-bin/soap/ws/quasi.cgi"/>
    </port>
</service>
```
CBS’ standard sequence data object

XSD

 Perl client

```xml
<xsd:complexType name="sequence">
  <xsd:sequence>
    <xsd:element name="id">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="[S-]"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="comment" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value=".*"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="seq" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="[A-Za-]+"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

my @SEQUENCEDATA = ( { id => "MY IDENTIFIER", comment => "My comment, which may contain spaces", seq => "ACGATGTGCGATCGATGATCGA", } );

my $request = $runOperation->{ parameters => { sequencedata => { sequence => [@SEQUENCEDATA] } } }
```

```perl
```
Granularity in data types (again)

Discussion: What are the benefits and the drawback of increased granularity?
Coffee break and Computer exercises