Accessing SOAP based Web Services using Perl

EMBRACE Workshop on Bioinformatics of Immunology

Peter Fischer Hallin, January 25, 2007
content

• Existing services (you already know a few from CBS)
• A brief introduction to the terms of Web Services
• Our deployment: Infrastructure, server-side software, queuing and logging
• From the clients view.
• The future: How will EMBRACE make a difference?
existing services
services

• We have known web services for years (home banking, ordering movie tickets and train tickets, submitting tax-forms, ordering pizza, comparing computer prices etc.)

• They have one thing in common: Build and designed to let humans interact with computers and databases over the web!

• A larger number of bioinformatic institutes have contributed by putting their bioinformatic programs online - mostly by using HTML forms. That is, you paste in (or upload) your input data, press submit and wait to retrieve your result. You interact with the computer.
Similar services are available from CBS at http://www.cbs.dtu.dk/services/, presented by category:

- Whole genome visualization and analysis
- Gene finding and splice sites
- Analysis of DNA microarray data
- Protein sorting
- Post-translational modifications of proteins, Immunological features
- Protein function and structures.
Similar services are available from CBS at http://www.cbs.dtu.dk/services/, presented by category:

- Whole genome visualization and analysis
- Gene finding and splice sites
- Analysis of DNA microarray data
- Protein sorting
- Post-translational modifications of proteins, Immunological features
- Protein function and structures.
existing prediction servers at CBS

• Highly used and cited
• Uniform and user-friendly web interfaces that allows the user to submit single jobs easily.
• Difficult to automate and to combine results from different services.
• The creator of a given services has no standard or definition to which he/she can conform. This leaves us with incompatible services.

solution: Web Services

Hopefully, the implementation will not invert the green and the red dots, making WS compliant services less sited, less user-friendly ;-)}
Web Services in 30sec

• Most Web Services are described by WSDL files (by some pronounced ‘wizdull’). Stands for Web Services Description Language.

• Concept: Each service will have a WSDL file. Once acquired, this file will describe all aspects of the service: what the services requires, what it provides, which functions (operations) it has and to which address (endpoint) the request should be directed.

• Definitions of input/output is declared either within the WSDL or as included files, in the format of XML Schema Definitions, XSDs.

• WSDL-files can be loaded by modules in Perl, C, Python and operations are imported as object functions. Code can easily be written which parses a response from one service and sends it as requests to another (if objects are compatible)
Things Web Services can offer you ... 

- Tight control over input and output objects via XSDs
- Overview of available functions (operations) defined in WSDLs (Web Services Description Language)
- Allow you to interact with the service from other application like Perl, C, Python etc.
- Allow you to write programs to join input and output of different services
CBS will work towards having all its prediction servers available also as SOAP based Web Services.
deployment
two templates for all services

• Creating templates into which all services will fit.

• Simple response/request services e.g. a database lookup. The message of the request is the result of the service. They are not queued and run per request.

• Asynchronous: Bigger jobs which takes longer time. Runs in a queue system.
normal vs. asynchronous jobs

Remember, that XSDs and WSDL does not allow to have different content of the same function. So having the same function return “job not ready yet” and then 5 minutes later return the answer is not an option!
The CBS WSDLs

• All Schema definitions of our services are placed in XSDs. Each WSDL contains references to two XSDs: One which is specific for the service and a common XSD which is shared among all WSDL (contains commonly used object like those used by the queue.

• Documentation is located within the WSDL in <documentation> tags and indexed by the web pages.
example: a WSDL for an rRNA prediction tool (RNAmmer)
example: a WSDL for an rRNA prediction tool (RNAmmer)

```xml
<?xml version="1.0"?>
  <documentation/>
  <wsdl:types>
    <message name="runServiceRequest"/>
    <message name="runServiceResponse"/>
    <message name="fetchResultRequest"/>
    <message name="fetchResultResponse"/>
    <message name="pollQueueRequest"/>
    <message name="pollQueueResponse"/>
    <portType name="WSRNAmmer_1_1a"/>
    <binding name="WSRNAmmerSoapBinding" type="tns:WSRNAmmer_1_1a"/>
    <service name="WSRNAmmer_1_1a"/>
  </wsdl:types>
</wsdl:definitions>
```
example: a WSDL for an rRNA prediction tool (RNAmmer)
example: a WSDL for an rRNA prediction tool (RNAmmer)
example: a WSDL for an rRNA prediction tool (RNAmmer)
example: a WSDL for an rRNA prediction tool (RNAmmer)
example: a WSDL for an rRNA prediction tool (RNAmmer)
prediction of rRNA genes: Alignment from secondary structure and construction of HMMs
RNAmer 1.1a Server

The RNAmer 1.1a server predicts 5s/8s, 16s/18s, and 23s/28s ribosomal RNA in full genome sequences. This page is the entry of the CBS Prediction Server for RNAmer. RNAmer is available also as a Web Service described by the following WSDL file. Please read the instructions on the RNAmer Web Services section.

SUBMISSION

Paste a single sequence or several sequences in FASTA format into the field below: Select kingdom of input sequences:

Bacteria

Submit a file in FASTA format directly from your local disk:

Submit | Clear fields

Restrictions:
RNAmmer 1.1a

Prediction of ribosomal RNA sub units

WSDL
Schema definitions
RNAmmer/RNAmmer_1_1a.wsdl
/common/ws_common_1_0.xsd
ws_rnammer_1_1a.xsd

We recommend that the first time users should load the WSDL file above to SoapUI and investigate the Web Services operations in that environment. SoapUI is a desktop application for inspecting, invoking, developing and functional/load/compliance testing of Web Services over HTTP. It can be downloaded free of charge from http://www.soapui.org/

Documentation

This Web Service predicts location of ribosomal RNA genes in full genome sequences by using Hidden Markov Models based on alignments from a highly curated dataset of structurally aligned sequences.

The input is one or more genomic sequence(s) as either one or more contigs. Each contig is submitted as one continuous string of DNA together with the sequences identifier.

The kingdom (either Bacteria, Archaea, or Eukaryotes) is specified once for each job that is submitted, using the abbreviations 'bac', 'arc', and 'euk'.

More information about the method can be found at:

http://www.cbs.dtu.dk/services/RNAmmer/
Queue and logs

- A queue system return job identifiers and let administrators control who can submit how much and how often, and to which services.
- A log system monitors sizes of input and output messages and which functions are being used.
- For internal usage, a Web Service described by WSDL and XSDs has been created which allows administrators to query both logs and queue.
on the clients view
We continue by looking at the prediction of rRNA genes. Submitting a genome sequence via the traditional interface will return a simple GFF output: A stringent format, which does not allow many additional output to be returned...
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:
Using soapUI it looks like this:

```
<soapenv:Envelope
  <soapenv:Body>
    <wsr:getResult>
      <job>
        <getProt>
          <mol>23s_rRNA</mol>
          <start>171731</start>
          <stop>174644</stop>
        </getProt>
      </job>
    </wsr:getResult>
  </soapenv:Body>
</soapenv:Envelope>
```

Response time: 684ms (20846 bytes)
<?xml version="1.0"?>
<xsd:schema xmlns:tns="http://www.cbs.dtu.dk/ws/ws-rnammer"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns:common="http://www.cbs.dtu.dk/ws/ws-common"
           xmlns:rnammer="http://www.cbs.dtu.dk/ws/ws-rnammer"
           targetNamespace="http://www.cbs.dtu.dk/ws/ws-rnammer"
           elementFormDefault="unqualified">
  <xsd:complexType name="runService">
    <xsd:sequence>
      <xsd:element name="parameters">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="kingdom">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="arc"/>
                  <xsd:enumeration value="bac"/>
                  <xsd:enumeration value="euk"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="sequences" type="common:sequences"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="expectedOutput">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="predictionTitle">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string"/>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="predictor">
              <xsd:complexType>
                <xsd:restriction base="xsd:string"/>
              </xsd:simpleType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
• XML is a richer format with few limitations. The author of the service is free to define any level of complexity in the XSD. With RNAmmer, we included most relevant output from HMMER.

• The soapUI method is without doubt less convenient. For humans. Todays exercise will demonstrate how Perl will handle exactly this task.
#!/usr/bin/perl

# Remember to install the latest version!
use SOAP::Lite;

# create SOAP services entry points
my $getseq = SOAP::Lite->service('http://www.cbs.dtu.dk/ws/GenomeAtlas/GenomeAtlas_3.0.wsdl');
my $rnammer = SOAP::Lite->service('http://www.cbs.dtu.dk/ws/RNAmer/RNAmer_1.1a.wsdl');

# input parameters for retrieving the sequence information
my $input = SOAP::Data->name(
  'parameters' => \SOAP::Data->value(
    SOAP::Data->name('accession' => "L43967")
  )
);

# method which will retrieve the sequences
my $sequences = $getseq->getSeq($input);

# ensure we are dealing with an array if only one entry is returned
if (ref ($sequences->[sequences]->[entry]) eq 'ARRAY') {
  @SEQ = @{$sequences->[sequences]->[entry]};
} else {
  @SEQ = ($sequences->[sequences]->[entry]);
}

printf STDERR "Got " . scalar(@SEQ) . " sequence(s)\n";

my @SEQUENCES;
foreach $sequence (@SEQ) {
  my ($seq, $ident) = ($sequence->[seq], $sequence->[ident]);
  push @SEQUENCES , ( SOAP::Data->name('entry' => \SOAP::Data->value(
    SOAP::Data->name('id' => $ident)
  )));
}

# input for finding rRNA genes
$input = SOAP::Data->name(
  'parameters' => \SOAP::Data->value(
  ...
'parameters' => \$SOAP::Data->value(
    \$SOAP::Data->name('kingdom' => "bac"),
    \$SOAP::Data->name('sequences' => \$SOAP::Data->value(@SEQUENCES))
);

# this method returns a jobid, which can be used to get the results later
my $job = $rnammer->runService($input);
my $jobid = \$SOAP::Data->name('job' => \$SOAP::Data->value(
    \$SOAP::Data->name('jobid' => $job->{jobid})
));

print STDERR "RNAmmer service launched with jobid: $job->{jobid}\n";

# set a loop for checking job status
my $job = $rnammer->pollQueue($jobid);

while ($job->{status} !~ /FINISHED|REJECTED|FAILED/) {
    print STDERR "The job is: $job->{status}\n";
    $job = $rnammer->pollQueue($jobid);
    sleep 1;
}

print STDERR "Job done, fetching results\n";

# when job is done fetch results
my $out = $rnammer->fetchResult($jobid);

# print Dumper($out);
if (ref ($out->{entries}) eq 'ARRAY') {
    @OUT = @{$out->{entries}};
} else {
    @OUT = ($out->{entries})
}

# print result
foreach $gene (@OUT) {
    print STDERR join("\t", ($gene->{sequenceEntry}, $gene->{start}, $gene->{stop}, $gene->{direction},
        $gene->{feature}, $gene->{score})), "\n";
}
the future
Web Services are here to stay

- The EMBRACE network will deliver ways for bioinformatic computers / grids to communicate using SOAP based Web Services.

- The major challenge: To ensure that the network partners agree on definitions of object types and content.

- In XSD, we could define a string holding a genomic sequence like this: `<xsd:restriction base="xsd:string">` - but it still takes a human to conclude that this is a genomic sequence and not a baking recipe! A true challenge remains: To let computers
Once the challenges are met, we can begin to exploit the power of Web Services: We will then be able to take a given genome sequence, protein sequence or an alignment and ask the network: “What can you do with this?”
Things Web Services can’t offer ...
Things Web Services can’t offer ...

- It will unlikely give the novice user an easier and more user-friendly way to use bioinformatic tools.
Things Web Services can’t offer ... 

- It will unlikely give the novice user an easier and more user-friendly way to use bioinformatic tools.

- It will not necessarily enable you to easily combine services from different organizations: XSDs must be somewhat compliant. Otherwise, you need to inspect the XSDs and split your XML messages before parsing them on to different services.
Things Web Services can’t offer ...

- It will unlikely give the novice user an easier and more user-friendly way to use bioinformatic tools.
- It will not necessarily enable you to easily combine services from different organizations: XSDs must be somewhat compliant. Otherwise, you need to inspect the XSDs and split your XML messages before parsing them on to different services.
- They can’t make coffee ...