Displaying External Data in Ensembl via DAS

The Ensembl genome browser can incorporate external data in the various displays via DAS (Distributed Annotation System). This document hopes to help explain the mechanism for bringing the two together and also give a tutorial on adding a demo data set. ProServer will be used here as the DAS server.

Prerequisites

1) ProServer installed

   See http://www.sanger.ac.uk/Software/analysis/proserver/ for instructions or see the tutorial for an example of how to set this up.

2) Ensembl genome browser

   This can be a locally installed copy (needed for permanent or global viewing of data) or access to http://www.ensembl.org (any changes made will only exist for that session and that user).

   See http://www.ensembl.org/Docs/EnsemblInstall_19_2.pdf or http://www.ensembl.org/Docs/EnsemblInstall_20_1.pdf for information on installing a local copy of the Ensembl genome browser depending on the version you wish to use.

Further Reading

DAS detailed information http://www.biodas.org/

Tutorial Examples Files

`Demo.ini`

`Demo.pm`

`Homo_sapiens.ini`

Setting up the ProServer DAS server, a tutorial

This is an example of setting up ProServer to serve out a demo set of data and testing that it works.

If you do not have Config::Inifiles (ver 2.38) then download and install from CPAN.

Download the server code:

```
$> cvs -d
:pserver:cvs@cvs.open-bio.org:/home/repository/biodas login
```

when prompted, the password is 'cvs'.

```
$> cvs -d
:pserver:cvs@cvs.open-bio.org:/home/repository/biodas
checkout Bio-Das2
```

Download the example code and to directory `Bio-Das2/eg/` we want to add `demo.ini`

```
$> cp ~/demo.ini Bio-Das2/eg/
```

If port 9000 is occupied on your machine please replace the line port =9000 with an alternative value.

To `Bio-Das2/Das/ProServer/SourceAdaptor` we want to add `demo.pm`

```
$> cp ~/demo.pm Bio-Das2/Das/ProServer/SourceAdaptor/
```

this file is the source adaptor for the demo data and returns correctly formatted data (this is explained in more detail later).

Make the make file (needed as directorys are searched to see what is included in the make).

```
$> perl Makefile.PL
```

Make the customised proserver.

```
$> make
```

Start the Server.

```
$> eg/proserver -c eg/demo.ini
```

Testing the DAS server

Can we see the DAS server from the web server?

Type http://DasServerMachine:9000/das/dsn into the location field of the web browser to list the available data sources that are available on the server hosted on DasServerMachine. Note that 9000 is the port. For this example we expect to see

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE DASDSN SYSTEM 'http://www.biodas.org/dtd/dasdsn.dtd' >
<DASDSN>
  <DSN>
    <SOURCE id="demo" version="1.0">demo</SOURCE>
    <MAPMASTER>http://DasServerMachine:9000/das/demo/</MAPMASTER>
    <DESCRIPTION>demo feature annotation</DESCRIPTION>
  </DSN>
</DASDSN>
```

Is the DAS server exporting the correct information?

Type into the browsers location field http://DasServerMachine:9000/das/demo/features?segment=10 to view the features that are available in the segment called segment10. This should look something like the following.

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE DASGFF SYSTEM "http://www.biodas.org/dtd/dasgff.dtd">
<DASGFF>
  <GFF version="1.01" href="http://DasServerMachine/das/demo/features">
    <SEGMENT id="10" version="1.0" start="1" stop="">
      <FEATURE id="10" label="10">
        <TYPE id="MADE_UP"  reference="no" subparts="no"
                                    superparts="no" >MADE_UP</TYPE>
        <METHOD id="MADE_UP">MADE_UP</METHOD>
        <START>50150000</START>
        <END>50200000</END>
        <ORIENTATION>0</ORIENTATION>
        <NOTE>Demo annotation 3 ??</NOTE>
    <LINK href="http://www.ensembl.org" >http://www.ensembl.org</LINK>
      </FEATURE>
    </SEGMENT>
  </GFF>
</DASGFF>
```

Test DAS server can be seen by Ensembl genome browser

Go to any ProteinView or GeneView page and click on 'Manage Sources' at ProteinDAS/GeneDAS Sources row. Now select 'NEW' and then 'more'. Now you can add your DAS server (include /das at the end and do not press return). Now click on 'DSN' list which will show you the DSN's available.

Choose demo and click apply.

If you choose http://ensemblserver/Homo_sapiens/protview?

peptide=CLAT_HUMAN to try this out on, then some of the demo data should now be viewable.

For above example a new row 'ProteinDAS: New_source' will appear with data in it and new features will appear in the Protein Features row.

Any user can add a DAS source to Ensembl via this method but the changes are not permanent and are for that user only. To make it permanent you must be able to edit the Ensembl server code.

Add data to the ProteinView/GeneView pages permanently

For the species you want to add the DAS server for, edit the ini file for that species,

e.g. `conf/Homo_sapiens.ini.`

To add data to ProteinView or GeneView pages the demo DAS data then after the line `[ENSEMBL_GENE_DAS_SOURCES]` add

```
Demo         = 1
[Demo]
dsn          = demo
url          = http://DasServerMachine:9000/das
type         = swissprot
authority    = http:/www.ensembl.org
on           = [ protview geneview ]
```

A brief description of each is given below.

`Demo`	set to 1 if Demo is to be included
`dsn`	set the data source name to be queried
`url`	the url the DAS server is accessed from
`type`	used to decide what segment names are passed
`authority`	url used for link with name
`on`	choose which pages the data should be used on

The allowed types are `swissprot`, `swissprot_acc`, `ensembl_gene` for branch 19-2 of the web code. Additionally `ensembl_peptide` and `ensembl_transcript` are available for version 20 and above.

Add data to the ContigView pages permanently

Again edit the <species>.ini file and after the line [ENSEMBL_INTERNAL_DAS_SOURCES] add

```
das_DEMO   = 1
[das_DEMO]
dsn        = demo
url        = http://DasServerMachine:9000/das
label      = demo data
caption    = demo data
col        = darkred
labelflag  = U
strand     = r
depth      = 6
group      = 1
on         = off
types      = [  ]
```

Here on states whether the data will be automatically displayed or not. In this example on is set to off so the DAS features will not be displayed until selected from the feature pulldown menu.

For these changes to take effect you will need remove the config.packed from the conf directory and restart the Ensembl genome browser.

Building DAS Server Source Adaptors for ProServer

What is needed from the DAS source adaptor and a review of the demonstration one given earlier is now explained.

The DAS source adaptor must have a minimum of three subroutines which are `init()`, `length()` and `build_features()`.

init()

`init()` defines what the Data Source Name (DSN) is and what is exported (what its capabilities are).

In the following example the dsn is demo and only data to be exported are features.

```
sub init {
  my $self                = shift;
  $self->{'dsn'}          = "demo";
  $self->{'capabilities'} = {'features' => '1.0',};
}
```

build_features()

`build_features()` must return an array of hashes. The XML layer is put on top by the parent class. In the demo example the data is obtained from an internal arrays at the top of the file.

The data looks like:-

```
my @id     = qw(CLAT_HUMAN CLAT_HUMAN CLAT_HUMAN 10 AC073366 );
my @id2    = qw(CLAT_HUMAN CLAT_HUMAN not_same 10 AC073366 );
my @names  = qw(PF00755 PF00755 CLAT_HUMAN 10 AC073366);
my @method = qw(description PFAM SCOP REPEAT BLAST);
my @starts = qw(30 100 50 50150000 1 );
my @ends   = qw(748 550 650 50200000 900);
my @types  = qw(typ1 typ2 typ3 typ4 typ5 typ6);
```

In the example given, types should be the type of thing that the data represents (e.g. genes, homology) with the methods describing how they were obtained (e.g. Genewise, Experimental).

Segment name should be used initially to filter the results then if defined, then start and end points should be used (this is to cut down on the data passed as ensembl code which will also perform start and end tests). Internal code will pass all the relevant segments to this subroutine.

```
sub build_features {
  my ($self, $opts) = @_;
  my $spid    = $opts->{'segment'};
  my $start   = $opts->{'start'};
  my $end     = $opts->{'end'};

  #
  #Create some dummy notes
  #
  my @notes=();
  for(my $i=0; $i <= $#id; $i++){
    $notes[$i] = "Demo annotation $i ??";
  }

  #create array of features to return;
  my @features = ();
  for(my $i=0; $i <= $#id; $i++){

    next if(($id[$i] ne $spid) ||
    (defined($start) and ($ends[$i] < $start or $starts[$i] > $end)));

     push @features, {
                      'id'     => $id2[$i],
                      'type'   => $types[$i],
                      'feature'=> $names[$i],
                     'method' => $method[$i],
                      'start'  => $starts[$i],
                      'end'    => $ends[$i],
                      'note'   => $notes[$i],
                      'link'   =>
                               'http://www.ensembl.org/Docs/enstour/',
                      'linktxt' => 'Tour',
                      };
  }

  return @features;
}


1;
```

length()

`length()` must return something. The return value + 1 is then added to the segment XML stop line

```
<SEGMENT id="10" version="1.0" start="1" stop="">
```

So the most basic subroutine would be

```
sub length {
  return 0;
}
```

Background on the Data needed

ProtView and GeneView

If the 'id' is not equal to the segment name then the feature will be displayed on the Protein Features row of the ProteinView page. This is shown here by the array `id2` having a value of `not_same`. The start and end are used here to display the annotation graphically.

If the segment name and id name do match then the annotation will be shown on the GeneDAS and ProteinDAS pages in the DAS row for that source. Start and ends are ignored in this case.

When editing the `species.ini` file and adding the demo DSN data it was specified that the type was swissprot. This means that all the segments which are passed to the build features where labeled by their Swissprot name, hence all the data was in terms of these.

ContigView

Here the data must be on a dna sequence e.g. chromosome 10 or AC073366 to be seen.

The start and ends are relative to the start of that segment.

Demo.ini

Consists of the following

```
[general]
prefork=5
maxclients=10
port=9000

[demo]
adaptor       = demo
state         = on
needed_arg    = okay
```

`prefork` states how many forks should be set of initially

`maxclients` states the maximum number of client

`port` states the port the service will be put on

`[demo]` states that the following will be for the DSN demo only

`adaptor` states what the perl adaptor `code .pm` is called

`state` states whether the service should be served or not

To pass information to the Source Adaptor (so that the same adaptor perl code can be used for many databases etc.) key value pairs are past by the ini file by having `x=y` in the example given above `needed_arg` is the key to access the value `okay`.

In the Source Adaptor the value is obtained via the call `$self->config->{'x'}` so to print the value in the source adaptor the call would be

```
print $self->config->{'needed_arg'};
```

This is useful for things like database name, user, password, host, file, etc.