Bio::DB::Das::Chado

As Middleware?

Scott Cain
Cold Spring Harbor Laboratory
Create the database

$ perl Makefile.PL
$ make
$ sudo make install
$ make load_schema
$ make prepdb     # now with Xenopus!
$ make ontologies # load rel, SO, featureprop

Then load some data...
Create some GFF from the spec

fake_chromosome example chromosome 1 15017 . . . ID=fake_chromosome;
   Name=fake_chromosome
fake_chromosome example gene 13691 14720 . + . ID=xfile;Name=xfile;
   Alias=mulder,scully;
   Note=A test gene for GMOD meeting
fake_chromosome example mRNA 13691 14720 . + . ID=xfile_mRNA;
   Parent=xfile
fake_chromosome example exon 13691 13767 . + . Parent=xfile_mRNA
fake_chromosome example exon 14687 14720 . + . Parent=xfile_mRNA
fake_chromosome example gene 12648 13136 . + . ID=x-men

And load it with the bulk loader:

$ gmod_bulk_load_gff3.pl -g sample.gff
...lots of output...
For kicks, set up GBrowse
<table>
<thead>
<tr>
<th>Name:</th>
<th>xfile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class:</td>
<td>gene:example</td>
</tr>
<tr>
<td>Type:</td>
<td>gene</td>
</tr>
<tr>
<td>Source:</td>
<td>example</td>
</tr>
<tr>
<td>Position:</td>
<td><code>fake_chromosome:13691..14720 (+ strand)</code></td>
</tr>
<tr>
<td>Length:</td>
<td>1030</td>
</tr>
<tr>
<td>Note:</td>
<td>A test gene for GMOD meeting</td>
</tr>
<tr>
<td>dbxref:</td>
<td>GFF_source:example</td>
</tr>
<tr>
<td>synonym:</td>
<td>mulder, scully, xfile</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parts:</th>
<th>Type: mRNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source:</td>
<td>example</td>
</tr>
<tr>
<td>Position:</td>
<td><code>fake_chromosome:13691..14720 (+ strand)</code></td>
</tr>
<tr>
<td>Length:</td>
<td>1030</td>
</tr>
<tr>
<td>dbxref:</td>
<td>GFF_source:example</td>
</tr>
<tr>
<td>synonym:</td>
<td>xfile_mRNA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parts:</th>
<th>Type: exon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source:</td>
<td>example</td>
</tr>
<tr>
<td>Position:</td>
<td><code>fake_chromosome:13691..13767 (+ strand)</code></td>
</tr>
<tr>
<td>Length:</td>
<td>77</td>
</tr>
<tr>
<td>dbxref:</td>
<td>GFF_source:example</td>
</tr>
<tr>
<td>synonym:</td>
<td>auto4, exon-auto4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parts:</th>
<th>Type: exon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source:</td>
<td>example</td>
</tr>
<tr>
<td>Position:</td>
<td><code>fake_chromosome:14687..14720 (+ strand)</code></td>
</tr>
<tr>
<td>Length:</td>
<td>34</td>
</tr>
<tr>
<td>dbxref:</td>
<td>GFF_source:example</td>
</tr>
<tr>
<td>synonym:</td>
<td>auto5, exon-auto5</td>
</tr>
</tbody>
</table>

`>xfile_mRNA class=mRNA:example position=fake_chromosome:13691..14720 (+ strand)
atacggttag tattcaggt tactgttttc gtacgtgta tcaccagcg tgtaggctgt ggaatcgac actggtgtaa`
Adaptor components

- Bio::DB::Das::Chado
  - Database connection object
- Bio::DB::Das::Chado::Segment
  - Object for any range of DNA
- Bio::DB::Das::Chado::Segment::Feature
  - Feature object
use Bio::DB::Das::Chado;

my $chado = Bio::DB::Das::Chado->new(
    -dsn => "dbi:Pg:dbname=test",
    -user=> "scott",
    -pass=> "" ) || die "no new chado";

my $gene_name = 'xfile';

my ($gene_fo) = $chado->get_features_by_name($gene_name);
Use some accessors

```perl
print "symbol: " . $gene_fo->display_name. "\n";
print "synonyms: " . join(', ', $gene_fo->synonyms). "\n";
print "description: " . $gene_fo->notes. "\n";
print "type: " . $gene_fo->type. "\n";

my ($mRNA) = $gene_fo->sub_SeqFeature();
my @exons = $mRNA->sub_SeqFeature();

for my $exon (@exons) {
    next unless ($exon->type->method eq 'exon');
    $exon_count++;
    print "exon$exon_count start: ". $exon->start. "\n";
    print "exon$exon_count end: " . $exon->end. "\n";
    $cds_seq .= $exon->seq->seq; #the first seq
} #returns a Bio::Seq object
```

my $gene_name = 'x-*';

my @genes = $chado->get_features_by_name(
  -name => $gene_name,
  -class=> 'gene' );

for my $gene (@genes) {
  print join("\t",
    $gene->feature_id,
    $gene->display_name,
    $gene->organism),"\n";
}
Advantages

- Comes 'for free' with GBrowse
- Uses 'familiar' BioPerl idioms, very similar to widely used Bio::DB::GFF (though with fewer methods).
Limitations

- No ability to write
- Incomplete implementation of Bio::DasI; just enough to make GBrowse work
- As an aside: found two bugs while working on this presentation (now fixed in cvs).
- Also, despite the name, has never been tested with a das server.
Conclusion

- Not suitable as a 'general' middleware layer
- However, it may be suitable for some applications, particularly if they are somehow similar to GBrowse or other uses of Bio::DB::GFF.