Modware: An Object-Oriented Perl Interface to Chado

Eric Just
Senior Bioinformatics Scientist
dictyBase: http://dictybase.org
Center for Genetic Medicine
Northwestern University
Outline

• Chado Features and Modware
• Architectural Overview
• Sample Problem
  – Insert Chromosome
  – Insert and Update Gene/mRNAs
  – Display Gene Reports
• Other Modware Highlights
• Coming soon
• Limitations
What is in the Feature Table? (the core of Chado)

• Chromosome
• Contig
• Gene
• mRNA
• Exon
• Lots of other things - See Sequence Ontology!
Modware Features

• Multiple Feature classes
  CHROMOSOME, GENE, MRNA, CONTIG
• Each class provides type specific methods
• Logic such as building exon structure of mRNA features is encapsulated
• Parent class Modware::Feature
  – Provides common methods
  – Abstract factory for various feature types
Architectural Overview

- Object-oriented Perl interface to Chado
- Built on top of Chado::AutoDBI
- Connection handled by GMOD
- Database transactions supported
- BioPerl used to represent and manipulate sequence and feature structure
- ‘Lazy’ evaluation
Create and Insert Chromosome

my $seq_io = new Bio::SeqIO(
    -file => "../data/fake_chromosome.txt",
    -format => 'fasta'
);

# Bio::SeqIO will return a Bio::Seq object which
# Modware uses as its representation
my $seq = $seq_io->next_seq();

my $reference_feature = new Modware::Feature(
    -type => 'chromosome',
    -bioperl => $seq,
    -description => "This is a test",
    -name => 'Fake',
    -source => 'GMOD 2007 Demo'
);

# Inserts chromosome into database
$reference_feature->insert();
Create and Insert a Gene

1) Enter the information about the following three novel genes, including the associated mRNA structures, into your database. Print the assigned feature_id for each inserted gene.

Gene Feature
  symbol: x-ray
  synonyms: none
mRNA Feature
  exon:
    start: 1703
    end: 1900
    strand: 1
    srcFeature_id:
      Id of genomic sample
Create and Insert a Gene

1) Enter the information about the following three novel genes, including the associated mRNA structures, into your database. Print the assigned feature_id for each inserted gene.

Gene Feature
symbol: x-men
synonyms: wolverine
mRNA Feature
exon_1:
start: 12648
end: 13136
strand: 1
srcFeature_id:
  Id of genomic sample
Create and Insert a Gene

1) Enter the information about the following three novel genes, including the associated mRNA structures, into your database. Print the assigned feature_id for each inserted gene.

Gene Feature
symbol: xfile
synonyms: mulder, scully
description: A test gene for GMOD meeting

mRNA Feature
exon_1:
  start: 13691
  end: 13767
  strand: 1
  srcFeature_id:
    Id of genomic sample
exon_2:
  start: 14687
  end: 14720
  strand: 1
  srcFeature_id:
    Id of genomic sample
Create and Insert a Gene

symbol: xfile
synonyms: mulder, scully
description: A test gene for GMOD meeting

my $gene_feature = new Modware::Feature(
    -type => 'gene',
    -name => 'xfile',
    -description => 'A test gene for GMOD meeting',
    -source => 'GMOD 2007 Demo'
);

$gene_feature->add_synonym( 'mulder' );
$gene_feature->add_synonym( 'scully' );

# inserts object into database
$gene_feature->insert();
print 'Inserted gene with feature_id:'.$gene_feature->feature_id()."\n";
Create mRNA BioPerl Object

exon_1:
  start: 13691
  end: 13767
  strand: 1
  srcFeature_id: Id of genomic sample

exon_2:
  start: 14687
  end: 14720
  strand: 1
  srcFeature_id: Id of genomic sample

# First, create exon features (using Bioperl)
my $exon_1 = new Bio::SeqFeature::Gene::Exon(
    -start   => 13691,
    -end     => 13767,
    -strand  => 1,
    -is_coding => 1
);

my $exon_2 = new Bio::SeqFeature::Gene::Exon(
    -start   => 14687,
    -end     => 14720,
    -strand  => 1,
    -is_coding => 1
);

# Next, create transcript feature to 'hold' exons (using Bioperl)
my $bioperl_mrna = new Bio::SeqFeature::Gene::Transcript();

# Add exons to transcript (using Bioperl)
$bioperl_mrna->add_exon( $exon_1 );
$bioperl_mrna->add_exon( $exon_2 );
Create and Insert mRNA

The BioPerl object holds the location information, but now we want to create a Modware object and link it to the gene as well as locate it on the chromosome.

```perl
# Now create Modware Feature to 'hold' bioperl object
my $mrna_feature = new Modware::Feature(
    -type => 'mRNA',
    -bioperl => $bioperl_mrna,
    -source => 'GMOD 2007 Demo',
    -reference_feature => $reference_feature
);

# Associate mRNA to gene (required for insertion)
$mrna_feature->gene($gene_feature);

# inserts object into database
$mrna_feature->insert();
```
Writing the Report

2) Retrieve and print the following report for gene xfile

symbol: xfile
synonyms: mulder, scully
description: A test gene for GMOD meeting
type: gene
exon1 start: 13691
exon1 end: 13767
exon2 start: 14687
exon2 end: 14720
>xfie cds
ATGCGGTAGTATCTCATGGTTACTGTTTCGCTACTGATATCACCACGCTGTAGGCTGT
GGAATCGAACAATGGTATTGTATAATGTTTGTGAATACACTGAGAAATAA

use Modware::Gene;
use GMODWriter;

my $xfile_gene = new Modware::Gene( -name => 'xfile' );
GMODWriter->Write_gene_report( $xfile_gene );
Writing the Report

package GMODWriter;
sub Write_gene_report {
    my ($self, $gene) = @_; 
    my $symbol = $gene->name();
    my @synonyms = @{$gene->synonyms()};
    my $syn_string = join ",", @synonyms;
    my $description = $gene->description();
    my $type = $gene->type();

    # get features associated with the gene that are of type 'mRNA'
    my ($mrna) = grep { $_->type() eq 'mRNA' } @{$gene->features()};

    # use bioperl method to get exons from mRNA
    my @exons = $mrna->bioperl->exons_ordered();

    # Modware will return a nice fasta file for you.
    my $fasta = $mrna->sequence( -type => 'cds', -format => 'fasta' );

    # Now print the actual report
    print "symbol: $symbol
";
    print "synonyms: $syn_string
";
    print "description: $description
";
    print "type: $type
";

    my $count = 0;
    foreach my $exon (@exons) {
        $count++;
        print "exon$\{$count\} start: ",$exon->start()."\n";
        print "exon$\{$count\} end: ",$exon->end()."\n";
    }
    print "$fasta";
} 

Updating a Gene Name

3) Update the gene xfile: change the name symbol to x-file and retrieve the changed record. Regenerate gene report

```perl
use Modware::Gene;
use Modware::DBH;
use GMODWriter;

eval{
    # get xfile gene
    my $xfile_gene = new Modware::Gene( -name => 'xfile' );

    # change the name
    $xfile_gene->name( 'x-file' );
    # write changes to database
    $xfile_gene->update();

    # we can use the original object if we want, but instead
    # we refetch from the database to 'prove' the name has been changed
    my $xfile_gene2 = new Modware::Gene( -name => 'x-file' );
    # use our GMODWriter package to write report for x-file
    GMODWriter->Write_gene_report( $xfile_gene2 );
};
if ($@){
    warn $@;
    new Modware::DBH->rollback();
}
```
Search and Display Results

4) Search for all genes with symbols starting with "x-*". With the results produce the following simple result list (organism will vary):

1323  x-file  Xenopus laevis
1324  x-men  Xenopus laevis
1325  x-ray  Xenopus laevis

```perl
use Modware::Gene;
use Modware::DBH;
use GMODWriter;

# find genes starting with 'x-
my $results = Modware::Search::Gene->Search_by_name( 'x-*' );

# write the search results
GMODWriter->Write_search_results( $results )
```
Search and Display Results

4) Search for all genes with symbols starting with "x-*". With the results produce the following simple result list (organism will vary):

1323  x-file  Xenopus laevis
1324  x-men  Xenopus laevis
1325  x-ray  Xenopus laevis

```perl
sub Write_search_results {
    my ($self, $itr) = @_; 

    # loop through iterator
    while ( my $gene = $itr->next() ) {
        # simply print the requested information
        print $gene->feature_id()."\t".$gene->name()."\t".$gene->organism_name()."\n";
    }
}
```
Delete a Gene

5) Delete the gene x-ray. Run the search and report again.

1323  x-file  Xenopus laevis
1324  x-men  Xenopus laevis

```perl
# get the xray gene
my $xray = new Modware::Gene( -name => 'x-ray' );

# set is_deleted = 1, this will 'hide' the gene from Searches
$xray->is_deleted(1);

# write change to database
$xray->update();

# find genes starting with 'x-
my $results = Modware::Search::Gene->Search_by_name( 'x-*' );

# write the search results
GMODWriter->Write_search_results( $results )
```
Other Modware Highlights

• Easy to write applications with Modware
• Extensible
• Available through Sourceforge
• Easy to install
• Large unit test coverage
• Current release 0.2-RC1
  – Works with GMOD’s latest release
  – Sample script demoed here are available
    • sample_scripts directory
Other Nice Things About Modware

http://gmod-ware.sourceforge.net/doc/
Coming Attractions

• Support for changing genomic sequence
• ncRNAs
• UTRs
• Ontology modules
• Phenotype Annotations
• Send us your ideas!
Limitations

• Does not have full flexibility of Chado
• Not enough users to get quality feedback
• Performance (?)
• Language dependent
Acknowledgments

dictyBase

• PIs
  – Rex Chisholm, PhD
  – Warren Kibbe, PhD

• Programmer
  – Sohel Merchant

• Curators
  – Petra Fey
  – Pascale Gaudet, PhD
  – Karen Pilcher

Other Groups

• Funding
  – NIH (NIGMS and NHGRI)

• GMOD
  – Scott Cain
  – Brian O’Connor
  – Everyone else

• BioPerl

• SGD
Why Modware Was Developed

• Each feature type requires different behavior
• Want to leave schema semantics out of application
• Want to leverage work done in BioPerl
• Re-use code developed for common use cases