Perl based Schema Abstraction Layer for CHADO

Bradley I. Arshinoff and Roy Welch
http://xanthusbase.org (xanthus.wikimods.org)
Syracuse University, NY

WHO WE ARE:

xanthusBase.org- MOD for the delta proteo bacterium *Myxococcus xanthus*

WikiMods.org- Collection of MODS for prokaryotes with small research scommunites. (Replaces xanthusBase.org)

July 30 2008- Launching wikimods

http://xanthus.wikimods.org
http://xeno.wikimods.org
http://demo.wikimods.org

API OVERVIEW (1): Basic operations

JUST 4 BASIC OPERATIONS: {add, del, get, set}

```
use Chadosal.pm qw/basic/;
my $feature = new CA_feature($featurename, 'gene', $organismID);

my @synonyms = $feature->get(CA_SYNONYMS); #get array
                                          #of gene synonyms
my @psynonyms = $feature->get(CA_PROTEINSYNONYMS);
$feature->set(CA_FMIN, 3000); #set start site for gene

shift @synonyms;
push (@synonyms, “genewithstartsiteat3000bp”);
$feature->set(CA_SYNONYMS, \@synonyms); #new set of
                                        #synonyms
```

API OVERVIEW (2): configuration

```
$feature->set_logging(1); #track revision history of any
                          #changes we make

$feature->set_autochildsearch(1); #search sub features (eg.
                                  #mRNA,  tRNA, rRNA)

$feature->set_returntype(CA_SIMPLE); #other option is
                                     #CA_CHADOOBJ
```

HOW IT WORKS

-> Each CA_object is described in a single perl document

-> Each table which has a relationship to the main object/table is described with a perl package in the main document. These packages contain table relationship information.

-> AutoDBI/DBI used as base object layer

->CHADO_CA reflexively examines document and builds master object with a set of handlers.

->A set of valid fields, represented by constants, are exported to user classes using Exporter package

EXTENDING THE API: Just complete a template.

Templates ε {one_to_one, one_to_many, many_to_many, many_to_many_linkonly}

```
#################################################################################################
################################## DATA CLASS: CHADOAC_Featureprop ##############################
#### WHEN USEING TEMPLATE COPY ALL CODE AND ONLY FILL IN TEMPLATE VARS. ######################

package XMODFeatureprop; ##TEMPLATE VAR, package: packageName
use strict; use XMODConfig; use base CHADO_CAGeneric'; no warnings 'redefine';

#TEMPLATE VAR: $relationship_type: Database table relationship type; Between parent table, this table and [linker table  (iff many_to_many )]
use constant relationship_type => XMODGeneric::REL_ONE_TO_MANY;

use constant mytype => "Chado::Featureprop";  #TEMPLAET VAR: underltying chado object
use constant child_valuecolumn => "value";       #TEMPLATE VAR: value field for return type of CA_SIMPLE
use constant child_cvtermcolumn => "type_id";  #TEMPLATE VAR: CVterm column
#TEMPLATE VAR: parenttype: The name of the Autodbi object which acts are the parent to this object
use constant parenttype => "Chado::Feature";

#TEMPLATE VAR: $linkertabletype. The name of the linker table. ONLY APPLICABLE IF relationship_type = REL_MANY_TO_MANY.
use constant linkertabletype => undef;

#TEMPLATE VAR: $linker_field: column name from primary object that links to parent object (or to linker table iff many_to_many)
use constant linker_field => 'feature_id';

use constant cvterms =>  ( #TEMPLATE VAR: %cvterms: FORMAT= [attribute name (in code)] ==> [CV name]::[cv term name]
   'PRODUCT|GENE_PRODUCT|PRODUCTS|GENE_PRODUCTS' => 'autocreated::product',
   'SELENOCYSTEINE|SEC' => 'autocreated::selenocysteine', #   'sec' => 'autocreated::sec',
   'TRANSLATION|CDS|PROTEIN_SEQUENCE|PROTEIN_SEQ|AA_SEQUENCE|AMINO_ACID_SEQUENCE' => 'Sequence Ontology::CDS',
     .
     .
     .
);

   #SUB: NEW. Copy this sub as-is when templating.
   sub new {      my ($class, @args) = @_;      my $self = {}; bless $self, $class;      $self->SUPER::new(@args);      return $self;  }
```