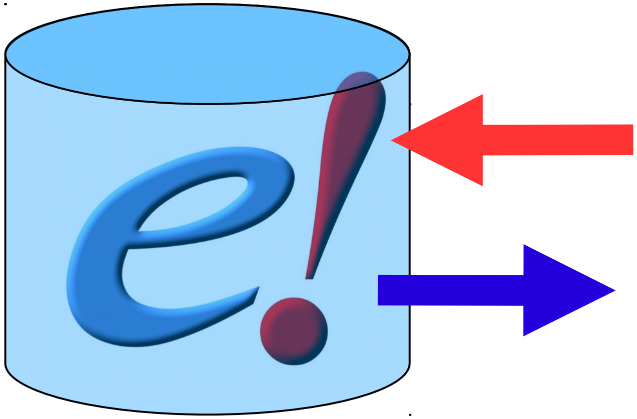
Ensembl Core API



EMBL-EBI

wellcome trust sanger institute

EMBL – European Bioinformatics Institute
Wellcome Trust Genome Campus
Hinxton, Cambridge, CB10 1SD, UK

Slides and Examples

Slides:

http://training.ensembl.org/events/2017/2017-02-09-APITaiwan

(https://goo.gl/ozhBs8)

Examples:

https://github.com/Ensembl/ensembl-presentation/tree/master/API/Core

Outline

a. Introduction

b. Data objects & object adaptors

c. Ensembl documentation

d. The Registry & Ensembl API script design

e. Coordinate systems & slices

f. Features

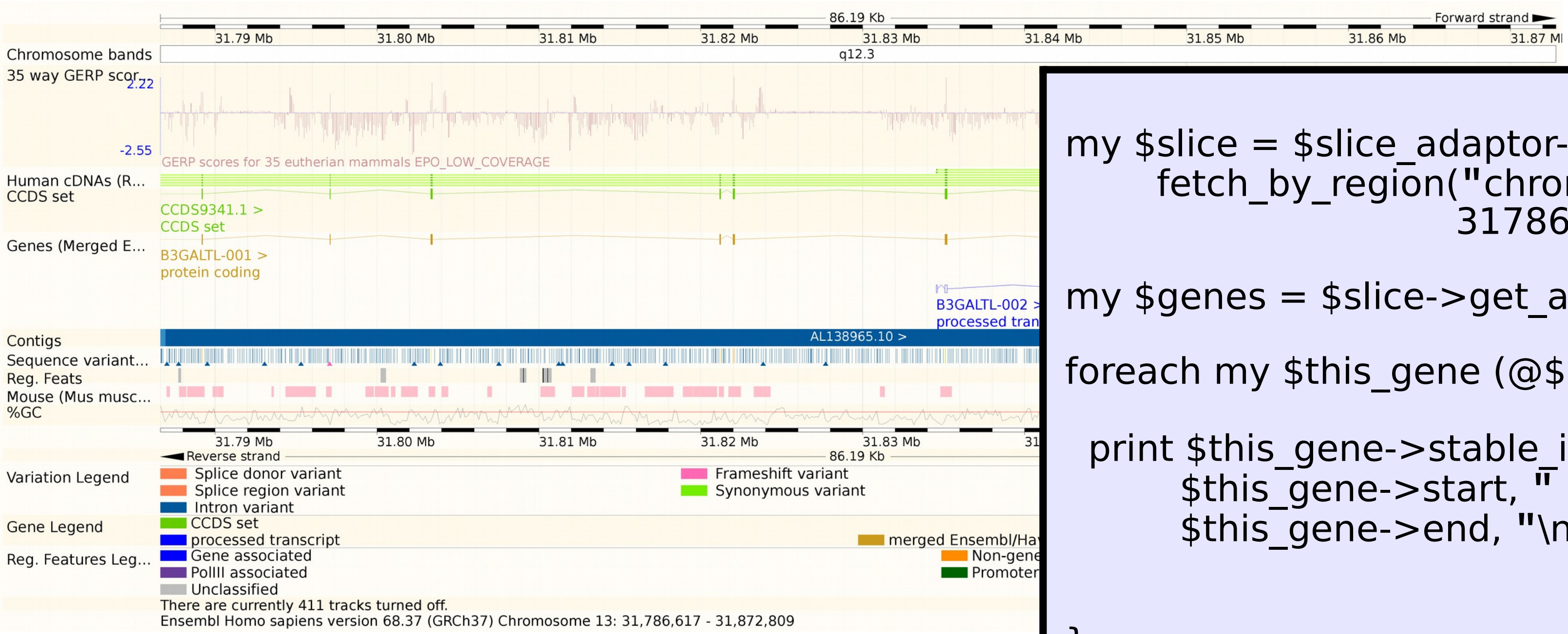
g. Genes, transcripts, exons & translations

h. External references

Slides: https://goo.gl/ozhBs8

Ensembl API

- Written in Object-Oriented Perl.
- Used to retrieve data from and to store data in Ensembl databases.
- Foundation for the Ensembl Pipeline and Ensembl Web interface.



Splice donor variant
Splice region variant
Intron variant
Frameshift variant
Synonymous variant
CCDS set
processed transcript
merged Ensembl/Ha
Gene associated
PolIII associated
Unclassified
Non-gene
Promoter

There are currently 411 tracks turned off.
Ensembl Homo sapiens version 68.37 (GRCh37) Chromosome 13: 31,786,617 - 31,872,809

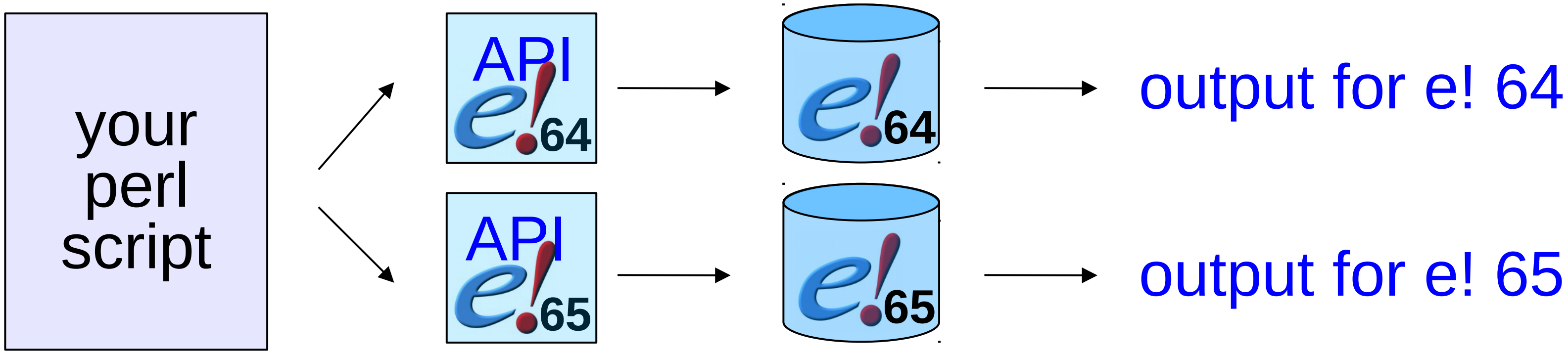
```
my $slice = $slice_adaptor->
       fetch_by_region("chromosome", "13",
                               31786617, 31872809);

my $genes = $slice->get_all_Genes();

foreach my $this_gene (@$genes) {

  print $this_gene->stable_id, ": ",
         $this_gene->start, "  -  ",
         $this_gene->end, "\n";


}
```

Why use an API?

- Uniform method of access to the data.
- Avoid writing the same thing twice: reusable in different systems.
- Reliable: lots of hard work, testing and optimisation already done.
- Insulates developers from underlying changes at a lower level (i.e. the database).

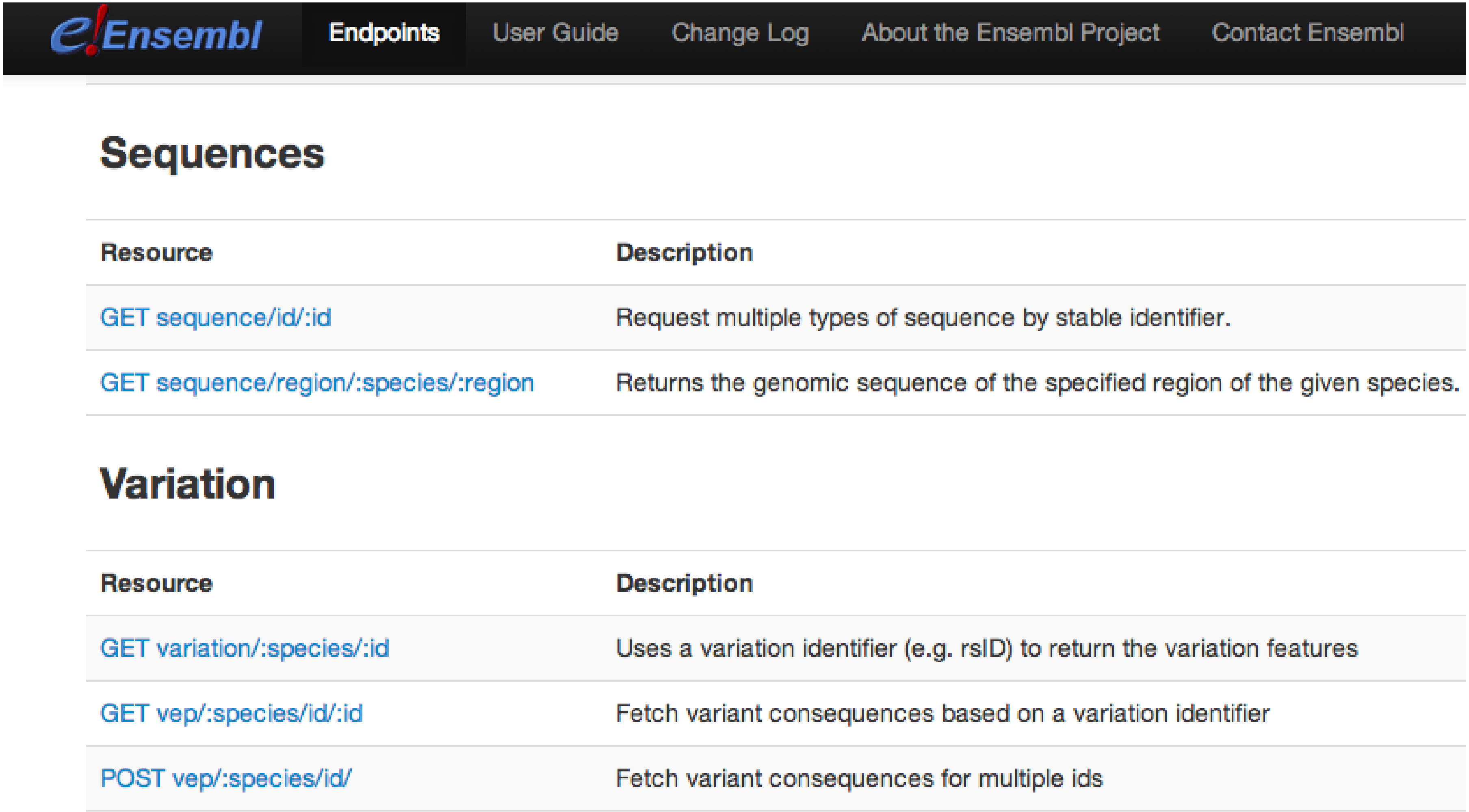
Using the correct API version

API version **must** match database version. Old scripts using the API *should* continue working with a newer API.



Run script ensembl/misc-scripts/ping_ensembl.pl to test if you can contact the Ensembl database server. The script will help you resolve issues with your setup.

An Alternative - REST



Sequences

Resource	Description
GET sequence/id/:id	Request multiple types of sequence by stable identifier.
GET sequence/region/:species/:region	Returns the genomic sequence of the specified region of the given species.

Variation

Resource	Description
GET variation/:species/:id	Uses a variation identifier (e.g. rsID) to return the variation features
GET vep/:species/id/:id	Fetch variant consequences based on a variation identifier
POST vep/:species/id/	Fetch variant consequences for multiple ids

http://rest.ensembl.org

Installing the Perl API from FTP

```
# cd to a location to install Ensembl to
mkdir src
cd src

# Get the latest API from FTP (always the live version) and BioPerl 1.6.1
wget ftp://ftp.ensembl.org/pub/ensembl-api.tar.gz
wget https://cpan.metacpan.org/authors/id/C/CJ/CJFIELDS/BioPerl-1.6.1.tar.gz

# untar both
tar zxvf ensembl-api.tar.gz
tar zxvf BioPerl-1.6.1.tar.gz

# open up .bashrc or .profile and add the following
PERL5LIB=${PERL5LIB}:${HOME}/src/bioperl-1.6.1
PERL5LIB=${PERL5LIB}:${HOME}/src/ensembl/modules
PERL5LIB=${PERL5LIB}:${HOME}/src/ensembl-compara/modules
PERL5LIB=${PERL5LIB}:${HOME}/src/ensembl-variation/modules
PERL5LIB=${PERL5LIB}:${HOME}/src/ensembl-functgenomics/modules
export PERL5LIB

# Checking your installation
perl $HOME/src/ensembl/misc-scripts/ping_ensembl.pl
```

Alternative Methods of Installation

- Git clone from https://github.com/Ensembl/
- Ensembl Virtual Machine
    - ftp://ftp.ensembl.org/pub/current_virtual_machine
    - 64bit only in OVA format
    - Available as a Vagrant box too

      vagrant init ensembl/ensembl; vagrant up --provider virtualbox

Outline

a. Introduction
b. Data objects & object adaptors
c. Ensembl documentation
d. The Registry & Ensembl API script design
e. Coordinate systems & slices
f. Features
g. Genes, transcripts, exons & translations
h. External references

Ensembl API – Object Types

We have two main object types in Ensembl API:

1. Data Objects

   Talk to a particular row in a data table, such as the BRCA2 gene to get (or set) information related to this gene only.

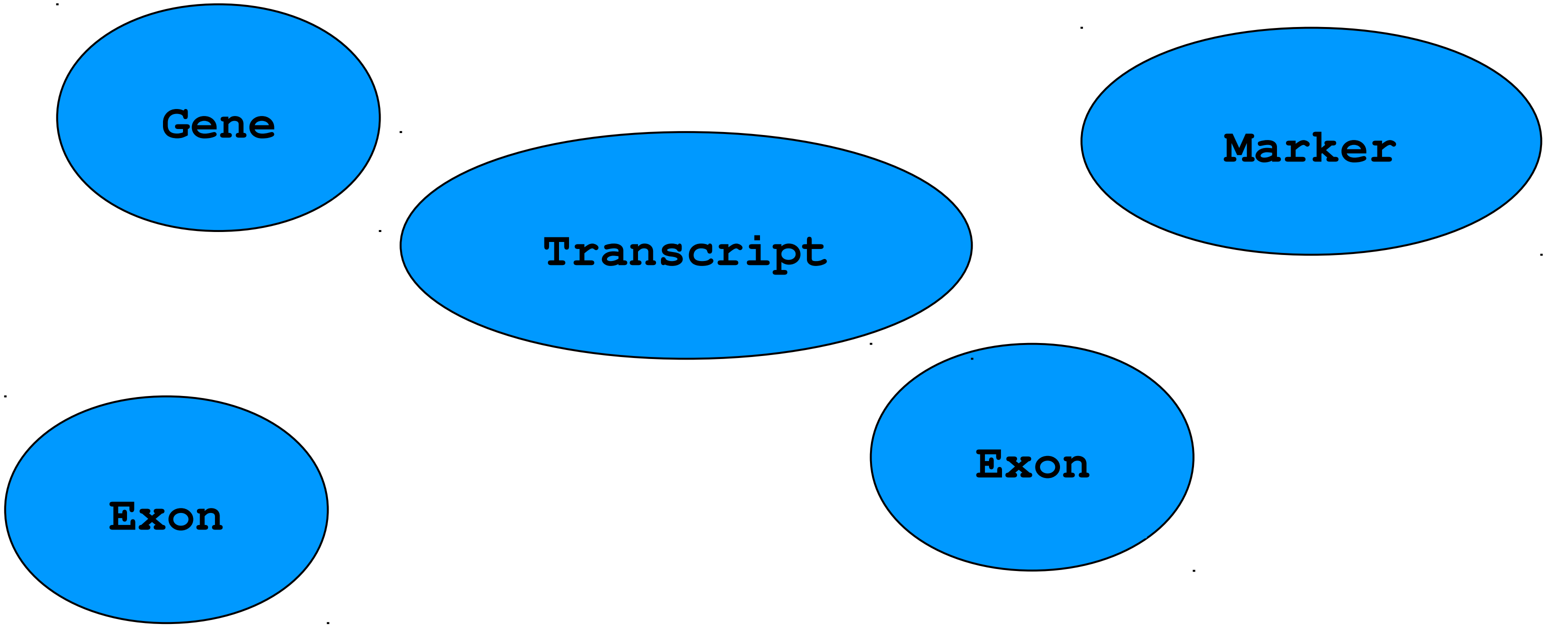
2. Object Adaptors

   Talk to a particular data table, such as the gene table to retrieve or store genes in the gene table.

Data Objects

Data objects model biological entities, e.g. genes, transcripts, exons…

A *Data Object* represents a piece of data that is (or can be) stored in the database.



Object Adaptors

Data Objects are retrieved from and stored in the database using ***Object Adaptors*.**

Each *Object Adaptor* is responsible for creating objects of only one particular type. For instance:

- The **Gene**Adaptor is used to fetch **Gene** objects
- The **Exon**Adaptor is used to fetch **Exon** objects

Object Adaptor fetch, *store*, and *remove* methods are used to retrieve, save, and delete information in the database.

Two types of methods:

- **fetch_by_....** returns 1 object (or undef)
- **fetch_all_by_...** returns a ref. to an array of objects (or ref. to an empty array)

Object methods and adaptors

- The API deals with objects representing database entities
- Adaptors are “**factories**” for generating objects
    - Adaptors are retrieved from the Registry

```
use strict;
Use warnings
use Bio::EnsEMBL::Registry;

my $reg = ‘Bio::EnsEMBL::Registry’;

$reg->load_registry_from_db(
    -host => ‘ensembldb.ensembl.org’,
    -user => ‘anonymous’
);

my $ga = $reg->get_adaptor(‘human’,   ‘core’,     ‘gene’);

...

my $gene= $ga->fetch_by_stable_id(‘ENSG00000139618’);
```

Use ‘strict’ and ‘warnings’

Change your host and port

species

group

object name

Object methods in Ensembl API

An Object has attributes and methods.

We avoid accessing object attributes directly, we use methods instead.

Methods are called using the “arrow” (->) operator:

```
my $exons = $gene->get_all_Exons();
```

Many methods can be used to either get or set an attribute value:

GET (no arg.):

```
my $gene_id = $gene->stable_id( );
```

SET (new value):

```
$gene->stable_id(“ENSG0000000123152”);
```

Object retrieval

- Using adaptors
  - **Fetch** object(s) according to some property e.g. name, location
    - “fetch_**all**_...” -> returns a list reference of items
    - “fetch_**by**_...” usually returns only 1 item
  - Check documentation which methods the adaptor provides

- Using API objects: e.g. **Slice, Gene, Transcript…**
  - **Get** other object(s) from an API object

    e.g. `$gene->`**`get_all_Transcripts()`**

    returns a list reference of Ensembl **Transcript** objects
  - Usually the object is written with a upper case in the method

    e.g. `get_all_Transcripts()`

Data Objects & Object Adaptors

```
# fetch a gene by its stable identifier using a gene
  adaptor

my $gene =
  $gene_adaptor->fetch_by_stable_id('ENSG00000139618');

# print out the name of a gene and its stable identifier

print $gene->external_name(), "\n";
print $gene->stable_id, "\n";
```

Ensembl Core modules

Name space for the modules:

- Object modules start with **Bio::EnsEMBL**
  - Bio::EnsEMBL::Gene for gene objects
  - Bio::EnsEMBL::Exon for exon objects
- ObjectAdaptors start with **Bio::EnsEMBL::DBSQL**
  - Bio::EnsEMBL::DBSQL::GeneAdaptor
  - Bio::EnsEMBL::DBSQL::ExonAdaptor

Outline

a. Introduction

b. Data objects & object adaptors

c. Ensembl documentation

d. The Registry & Ensembl API script design

e. Coordinate systems & slices

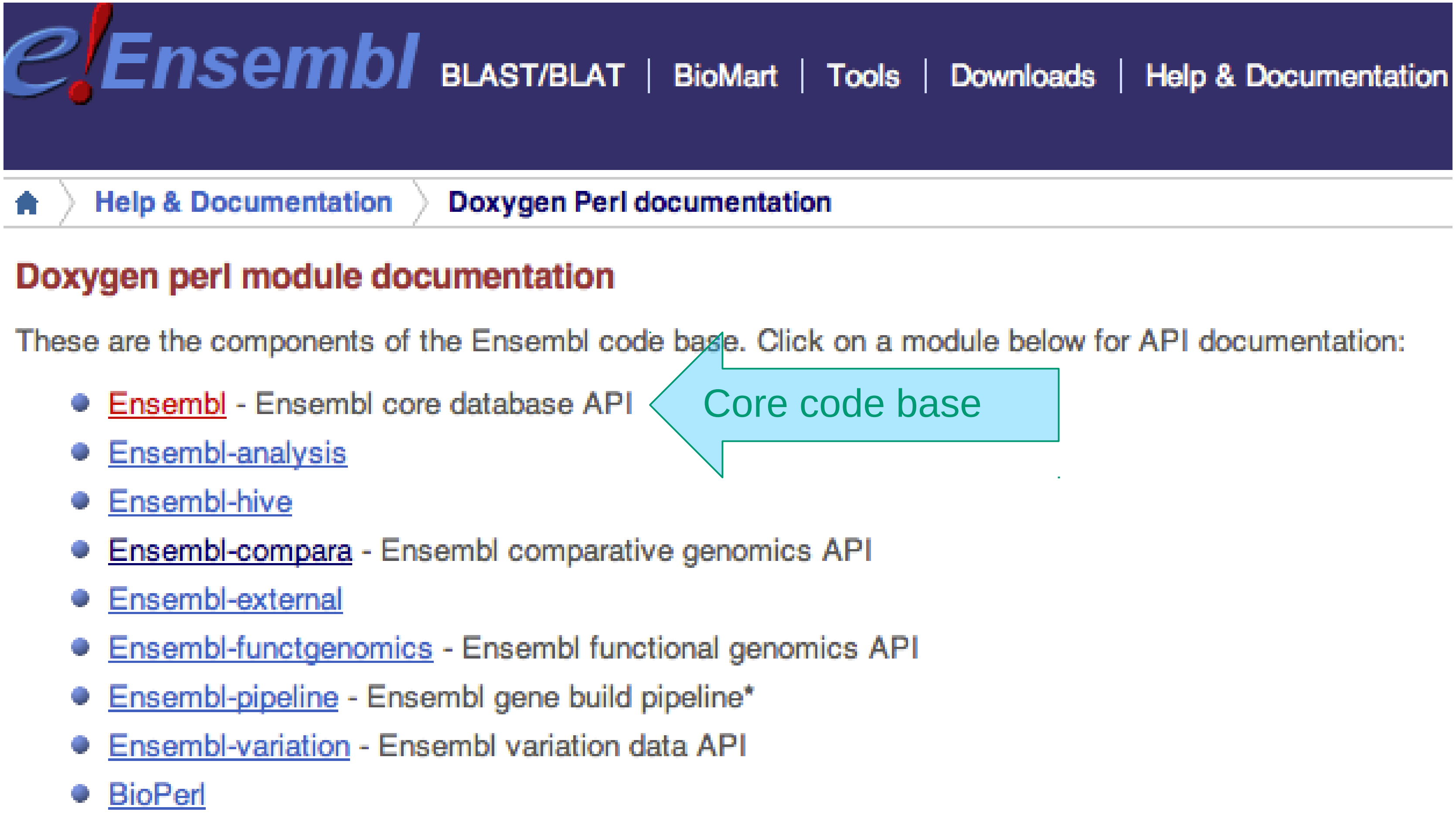
f. Features

g. Genes, transcripts, exons & translations

h. External references

Ensembl API Documentation (1)

Go to http://www.ensembl.org/info/docs/Doxygen/index.html


e!Ensembl
BLAST/BLAT | BioMart | Tools | Downloads | Help & Documentation
Help & Documentation
Doxygen Perl documentation

Doxygen perl module documentation

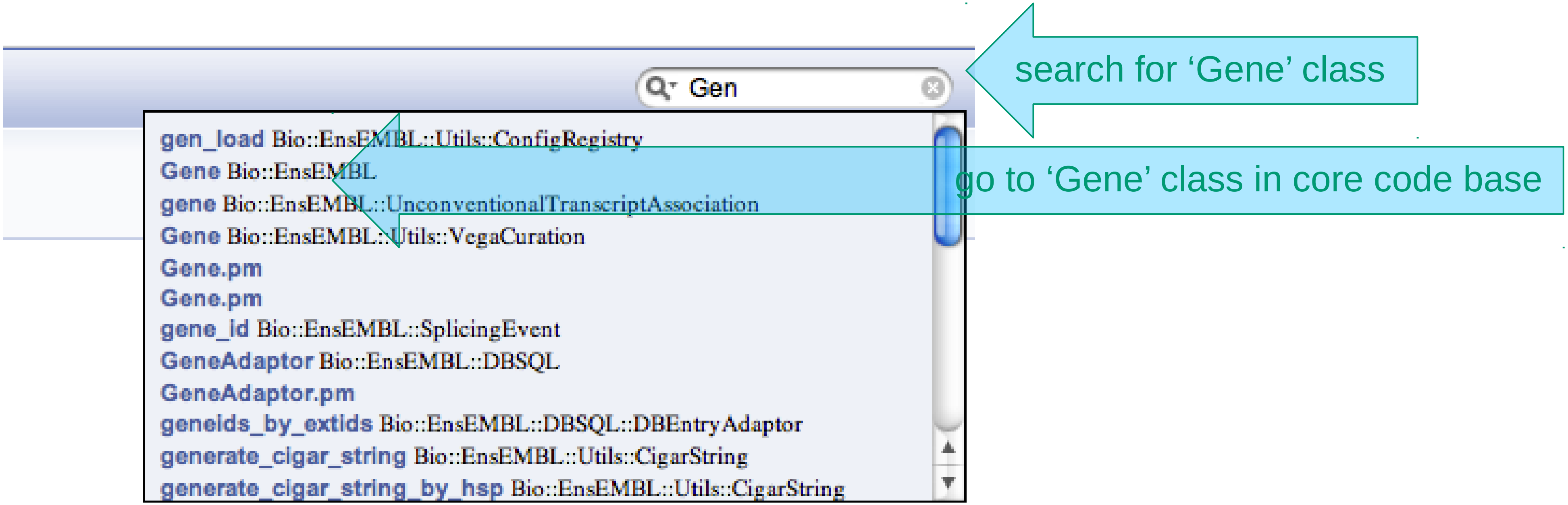
These are the components of the Ensembl code base. Click on a module below for API documentation:

- Ensembl - Ensembl core database API
- Ensembl-analysis
- Ensembl-hive
- Ensembl-compara - Ensembl comparative genomics API
- Ensembl-external
- Ensembl-functgenomics - Ensembl functional genomics API
- Ensembl-pipeline - Ensembl gene build pipeline*
- Ensembl-variation - Ensembl variation data API
- BioPerl

Core code base

Ensembl API Documentation (2)

Search for classes or methods



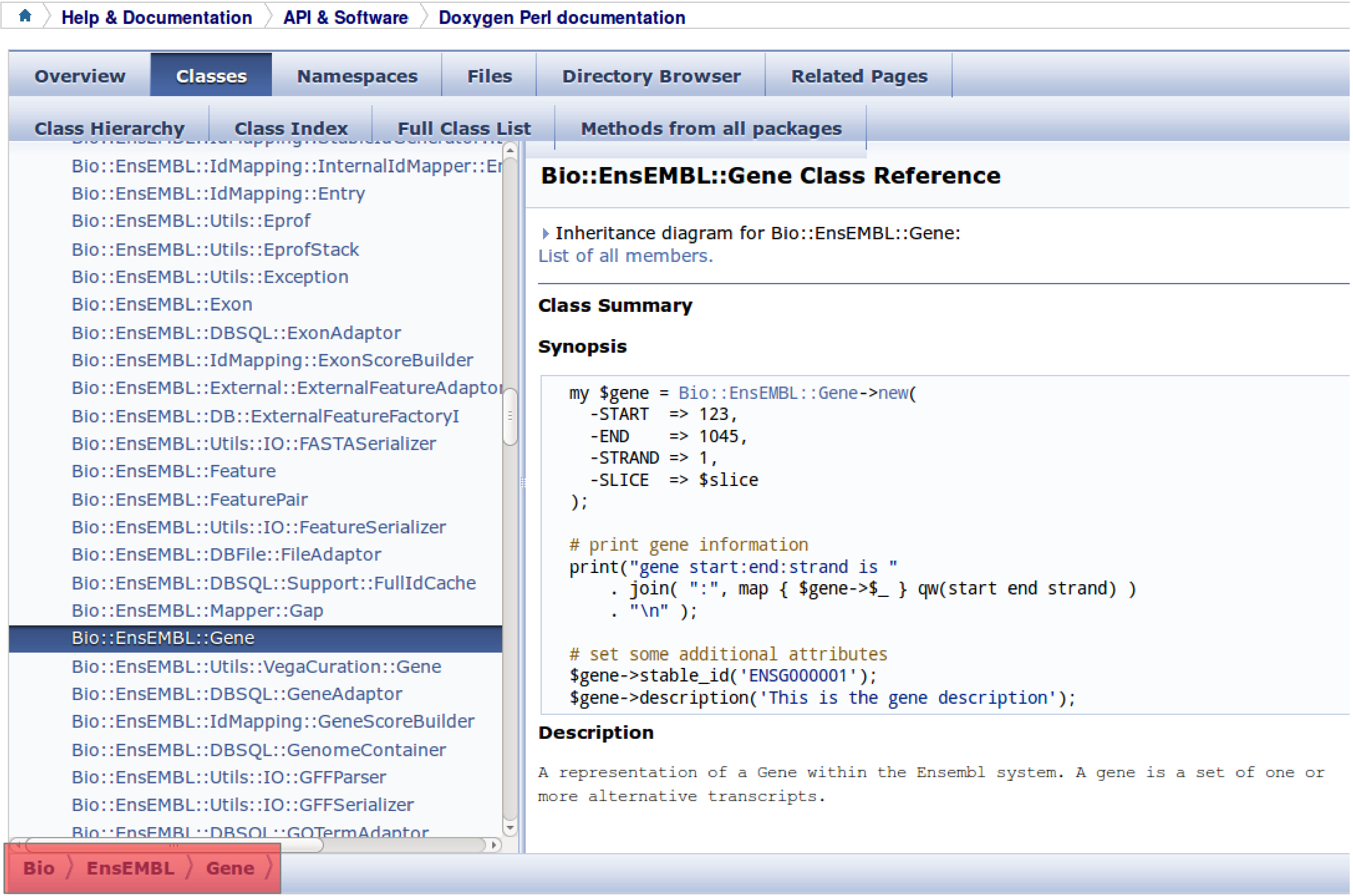
gen_load Bio::EnsEMBL::Utils::ConfigRegistry
Gene Bio::EnsEMBL
gene Bio::EnsEMBL::UnconventionalTranscriptAssociation
Gene Bio::EnsEMBL::Utils::VegaCuration
Gene.pm
Gene.pm
gene_id Bio::EnsEMBL::SplicingEvent
GeneAdaptor Bio::EnsEMBL::DBSQL
GeneAdaptor.pm
geneids_by_extids Bio::EnsEMBL::DBSQL::DBEntryAdaptor
generate_cigar_string Bio::EnsEMBL::Utils::CigarString
generate_cigar_string_by_hsp Bio::EnsEMBL::Utils::CigarString

search for ‘Gene’ class

go to ‘Gene’ class in core code base

Ensembl API Documentation (3)

Breadcrumbs help you keep track of which code base you’re in.


Help & Documentation
API & Software
Doxygen Perl documentation
Overview
Classes
Namespaces
Files
Directory Browser
Related Pages
Class Hierarchy
Class Index
Full Class List
Methods from all packages
Bio::EnsEMBL::IdMapping::InternalIdMapper::Er
Bio::EnsEMBL::IdMapping::Entry
Bio::EnsEMBL::Utils::Eprof
Bio::EnsEMBL::Utils::EprofStack
Bio::EnsEMBL::Utils::Exception
Bio::EnsEMBL::Exon
Bio::EnsEMBL::DBSQL::ExonAdaptor
Bio::EnsEMBL::IdMapping::ExonScoreBuilder
Bio::EnsEMBL::External::ExternalFeatureAdaptor
Bio::EnsEMBL::DB::ExternalFeatureFactoryI
Bio::EnsEMBL::Utils::IO::FASTASerializer
Bio::EnsEMBL::Feature
Bio::EnsEMBL::FeaturePair
Bio::EnsEMBL::Utils::IO::FeatureSerializer
Bio::EnsEMBL::DBFile::FileAdaptor
Bio::EnsEMBL::DBSQL::Support::FullIdCache
Bio::EnsEMBL::Mapper::Gap
Bio::EnsEMBL::Gene
Bio::EnsEMBL::Utils::VegaCuration::Gene
Bio::EnsEMBL::DBSQL::GeneAdaptor
Bio::EnsEMBL::IdMapping::GeneScoreBuilder
Bio::EnsEMBL::DBSQL::GenomeContainer
Bio::EnsEMBL::Utils::IO::GFFParser
Bio::EnsEMBL::Utils::IO::GFFSerializer
Bio::EnsEMBL::DBSQL::GOTermAdaptor
Bio
EnsEMBL
Gene

Bio::EnsEMBL::Gene Class Reference

Inheritance diagram for Bio::EnsEMBL::Gene:

List of all members.

Class Summary

Synopsis

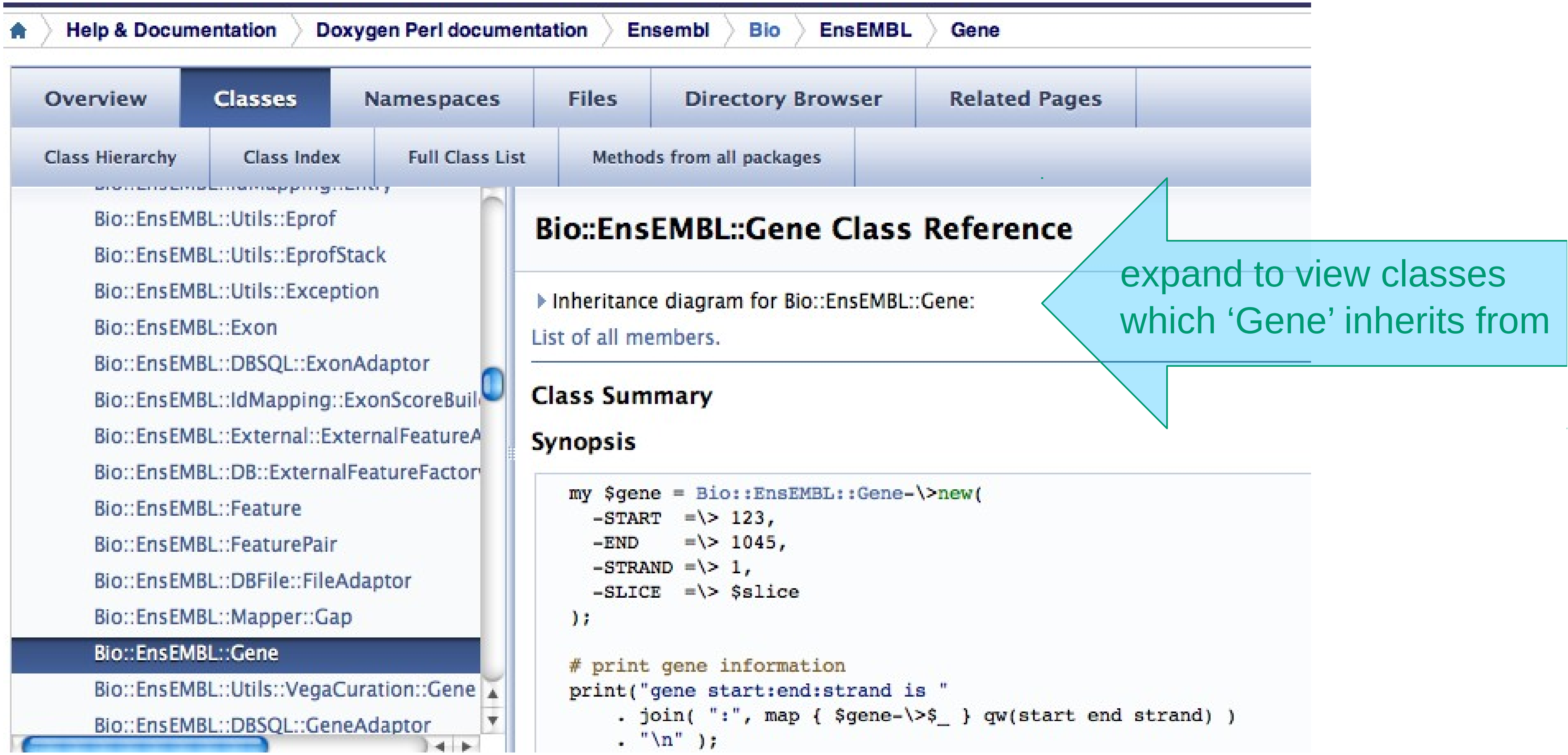
```
my $gene = Bio::EnsEMBL::Gene->new(
  -START  => 123,
  -END    => 1045,
  -STRAND => 1,
  -SLICE  => $slice
);

# print gene information
print("gene start:end:strand is "
    . join( ":", map { $gene->$_ } qw(start end strand) )
    . "\n" );

# set some additional attributes
$gene->stable_id('ENSG000001');
$gene->description('This is the gene description');
```

Description

A representation of a Gene within the Ensembl system. A gene is a set of one or more alternative transcripts.

Ensembl API Documentation (4)

Classes can *inherit* attributes and methods from other classes. For instance a Gene is also a Feature as it can be located on a region of DNA so it inherits from the **Bio::EnsEMBL::Feature** object.



Bio::EnsEMBL::Utils::Eprof
Bio::EnsEMBL::Utils::EprofStack
Bio::EnsEMBL::Utils::Exception
Bio::EnsEMBL::Exon
Bio::EnsEMBL::DBSQL::ExonAdaptor
Bio::EnsEMBL::IdMapping::ExonScoreBuil
Bio::EnsEMBL::External::ExternalFeatureA
Bio::EnsEMBL::DB::ExternalFeatureFactor
Bio::EnsEMBL::Feature
Bio::EnsEMBL::FeaturePair
Bio::EnsEMBL::DBFile::FileAdaptor
Bio::EnsEMBL::Mapper::Gap
Bio::EnsEMBL::Gene
Bio::EnsEMBL::Utils::VegaCuration::Gene
Bio::EnsEMBL::DBSQL::GeneAdaptor

Bio::EnsEMBL::Gene Class Reference

▸ Inheritance diagram for Bio::EnsEMBL::Gene:

List of all members.

Class Summary

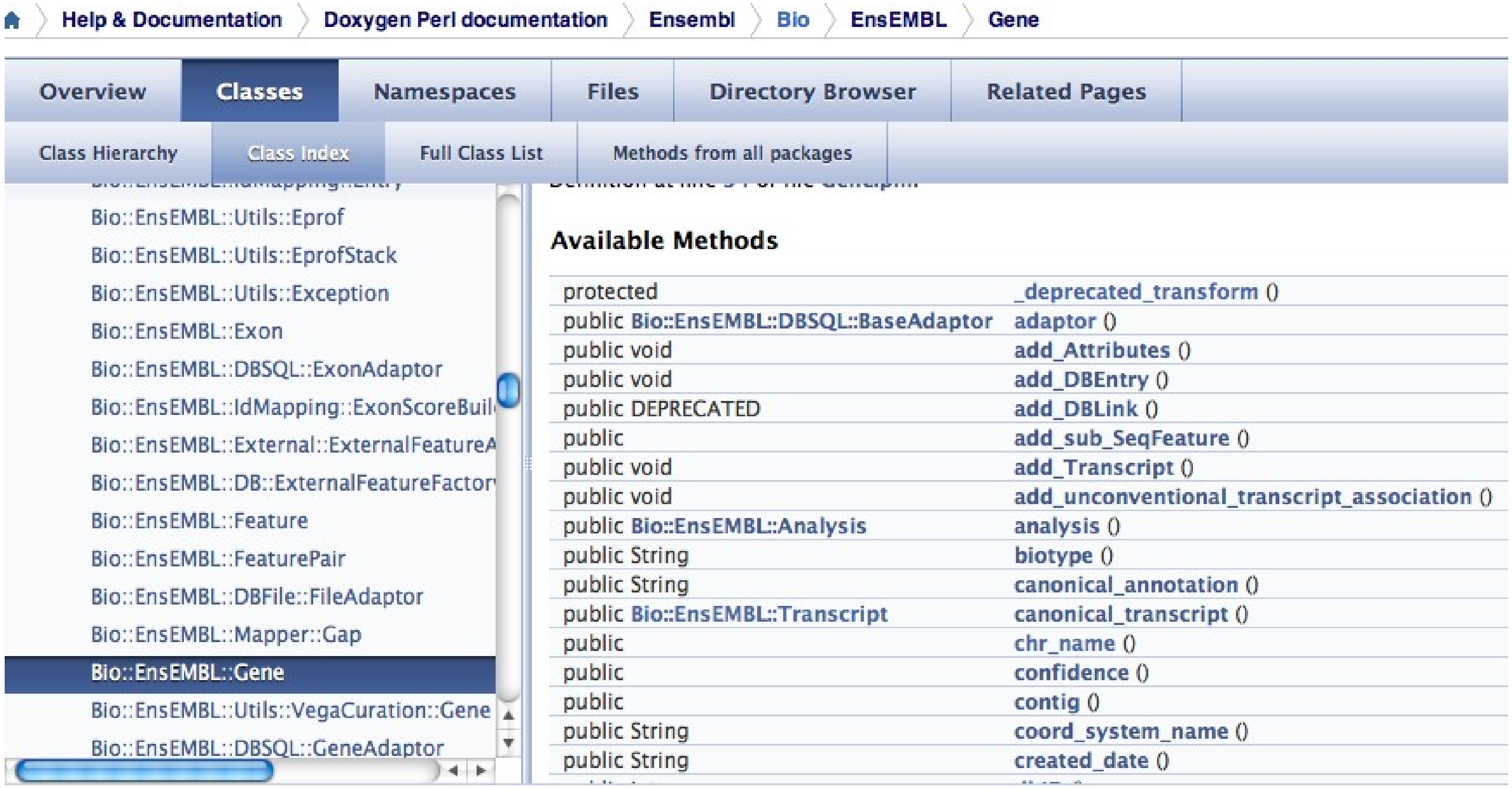
Synopsis

```
my $gene = Bio::EnsEMBL::Gene-\>new(
  -START  =\> 123,
  -END    =\> 1045,
  -STRAND =\> 1,
  -SLICE  =\> $slice
);

# print gene information
print("gene start:end:strand is "
    . join( ":", map { $gene-\>$_ } qw(start end strand) )
    . "\n" );
```

expand to view classes which ‘Gene’ inherits from

Ensembl API Documentation (5)

Scroll down to see a list of available methods in the ‘Gene’ class. Can you find a method to retrieve all transcripts for a gene? Where possible, this list shows the returned data types for each method.



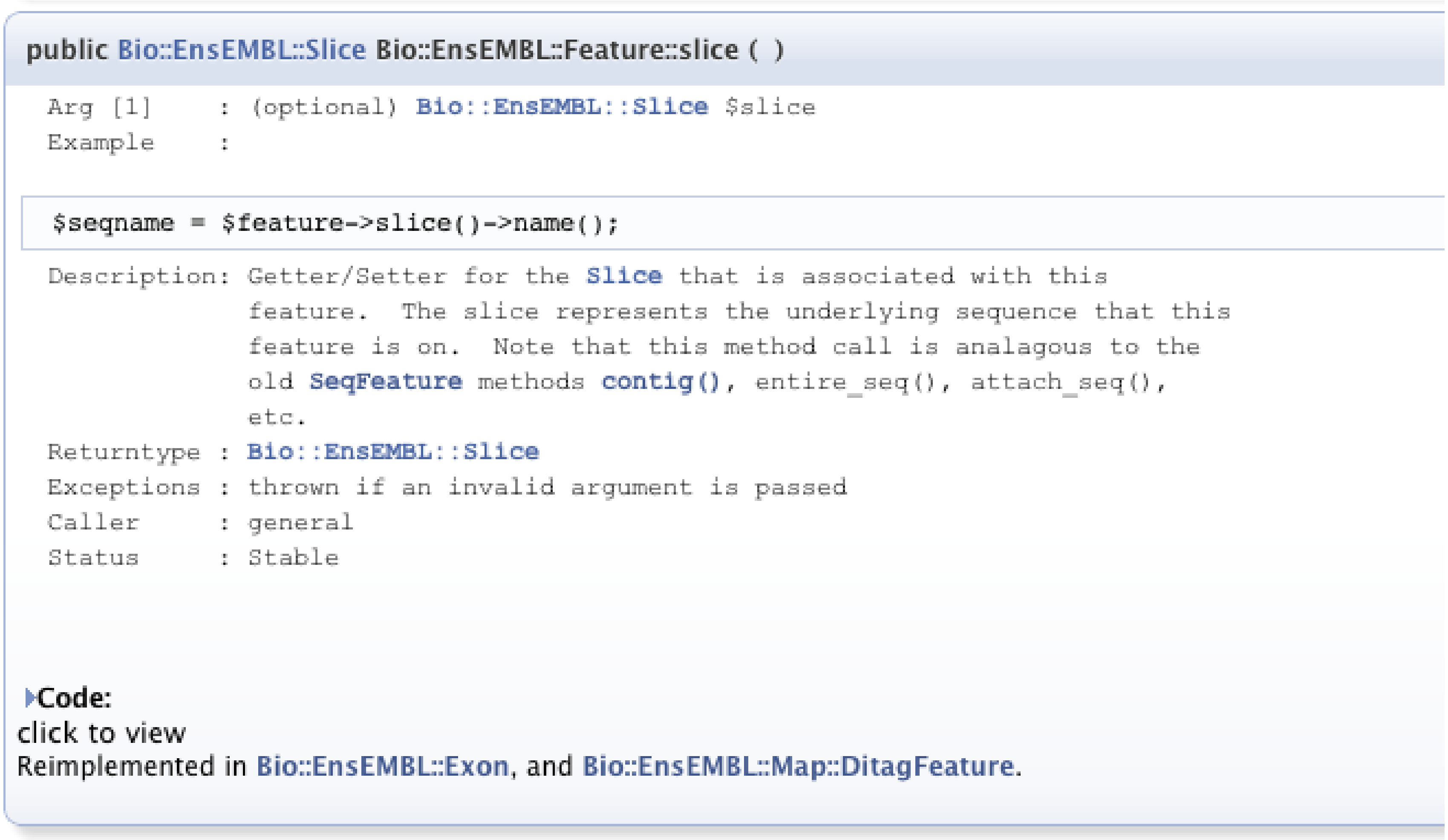
Bio::EnsEMBL::Utils::Eprof
Bio::EnsEMBL::Utils::EprofStack
Bio::EnsEMBL::Utils::Exception
Bio::EnsEMBL::Exon
Bio::EnsEMBL::DBSQL::ExonAdaptor
Bio::EnsEMBL::IdMapping::ExonScoreBuil
Bio::EnsEMBL::External::ExternalFeatureA
Bio::EnsEMBL::DB::ExternalFeatureFactor
Bio::EnsEMBL::Feature
Bio::EnsEMBL::FeaturePair
Bio::EnsEMBL::DBFile::FileAdaptor
Bio::EnsEMBL::Mapper::Gap
Bio::EnsEMBL::Gene
Bio::EnsEMBL::Utils::VegaCuration::Gene
Bio::EnsEMBL::DBSQL::GeneAdaptor

Available Methods

protected	_deprecated_transform ()
public Bio::EnsEMBL::DBSQL::BaseAdaptor	adaptor ()
public void	add_Attributes ()
public void	add_DBEntry ()
public DEPRECATED	add_DBLink ()
public	add_sub_SeqFeature ()
public void	add_Transcript ()
public void	add_unconventional_transcript_association ()
public Bio::EnsEMBL::Analysis	analysis ()
public String	biotype ()
public String	canonical_annotation ()
public Bio::EnsEMBL::Transcript	canonical_transcript ()
public	chr_name ()
public	confidence ()
public	contig ()
public String	coord_system_name ()
public String	created_date ()

Ensembl API Documentation (6)

Clicking on a method takes you to a description with associated arguments, return types and exceptions.



public Bio::EnsEMBL::Slice Bio::EnsEMBL::Feature::slice ( )

```
Arg [1]     : (optional) Bio::EnsEMBL::Slice $slice
Example     :
```

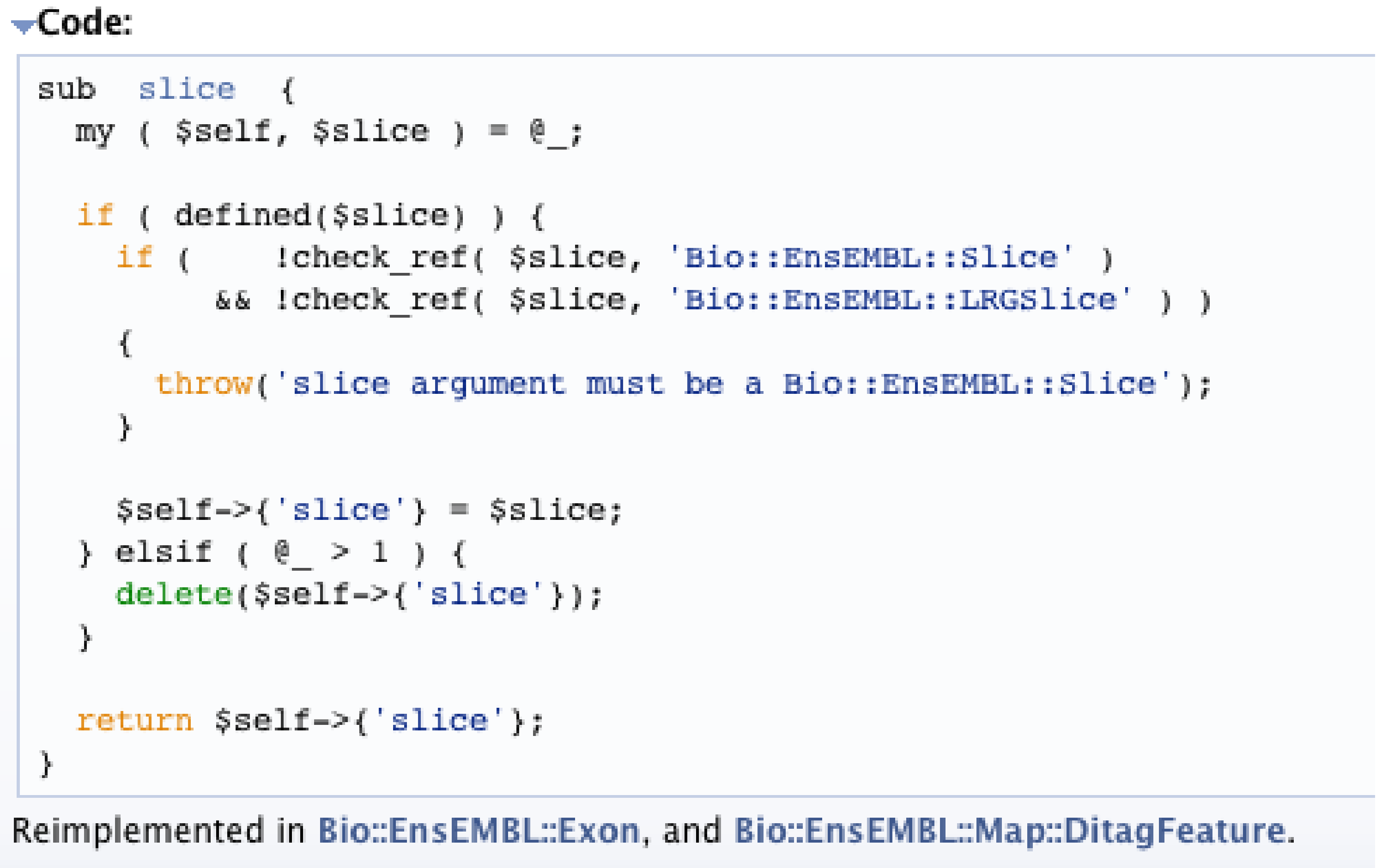
```
$seqname = $feature->slice()->name();
```

```
Description: Getter/Setter for the Slice that is associated with this
             feature.  The slice represents the underlying sequence that this
             feature is on.  Note that this method call is analagous to the
             old SeqFeature methods contig(), entire_seq(), attach_seq(),
             etc.
Returntype : Bio::EnsEMBL::Slice
Exceptions : thrown if an invalid argument is passed
Caller     : general
Status     : Stable
```

Code:
click to view
Reimplemented in Bio::EnsEMBL::Exon, and Bio::EnsEMBL::Map::DitagFeature.

Ensembl API Documentation (7)

Clicking on **Code:** shows the method's implementation



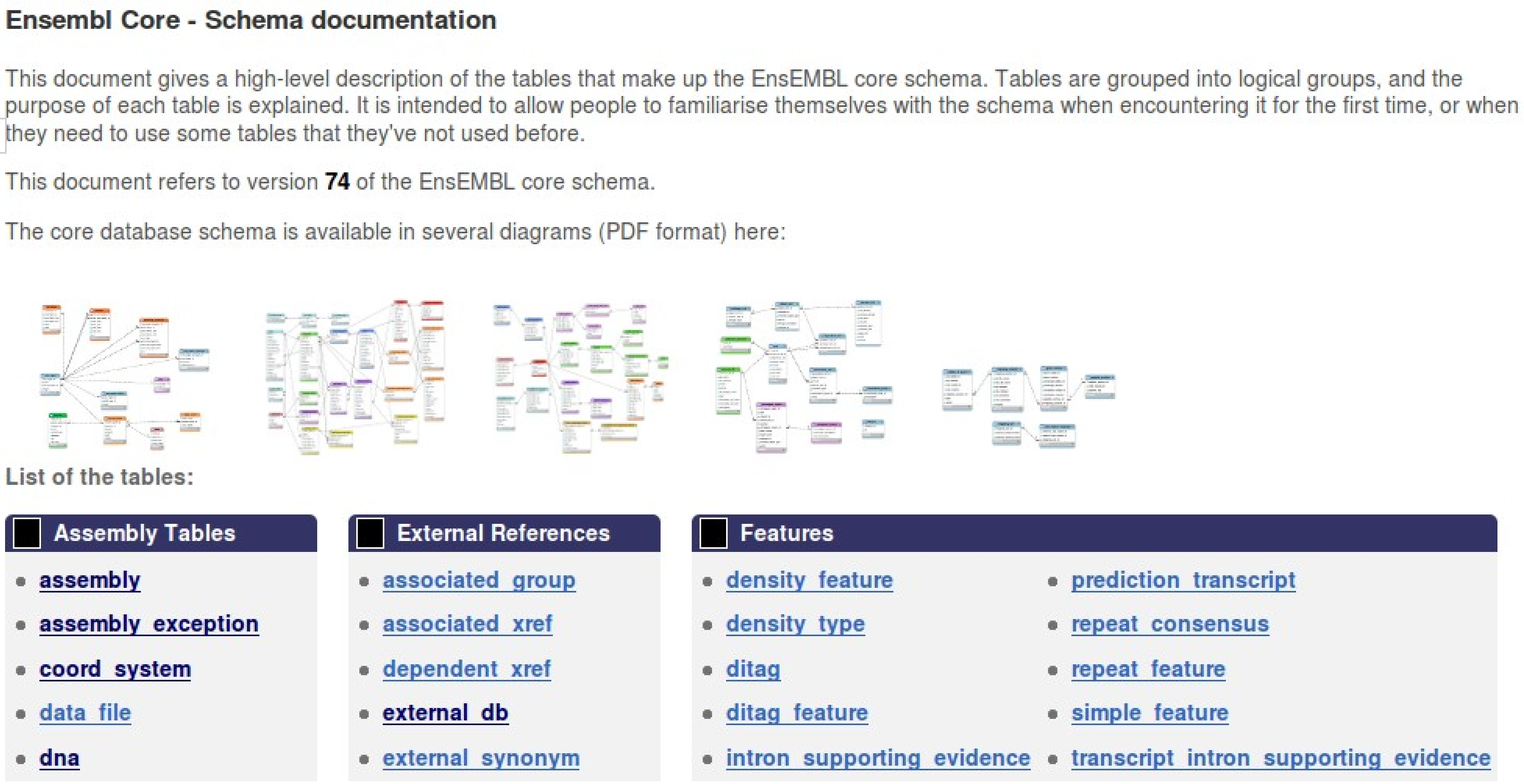
Code:

```
sub  slice  {
  my ( $self, $slice ) = @_;

  if ( defined($slice) ) {
    if (     !check_ref( $slice, 'Bio::EnsEMBL::Slice' )
         && !check_ref( $slice, 'Bio::EnsEMBL::LRGSlice' ) )
    {
      throw('slice argument must be a Bio::EnsEMBL::Slice');
    }

    $self->{'slice'} = $slice;
  } elsif ( @_ > 1 ) {
    delete($self->{'slice'});
  }

  return $self->{'slice'};
}
```

Reimplemented in Bio::EnsEMBL::Exon, and Bio::EnsEMBL::Map::DitagFeature.

Ensembl Core DB Documentation (1)

Go to http://www.ensembl.org/info/docs/api/core/core_schema.html



Ensembl Core - Schema documentation

This document gives a high-level description of the tables that make up the EnsEMBL core schema. Tables are grouped into logical groups, and the purpose of each table is explained. It is intended to allow people to familiarise themselves with the schema when encountering it for the first time, or when they need to use some tables that they've not used before.

This document refers to version **74** of the EnsEMBL core schema.

The core database schema is available in several diagrams (PDF format) here:

List of the tables:

Assembly Tables

- assembly
- assembly_exception
- coord_system
- data_file
- dna

External References

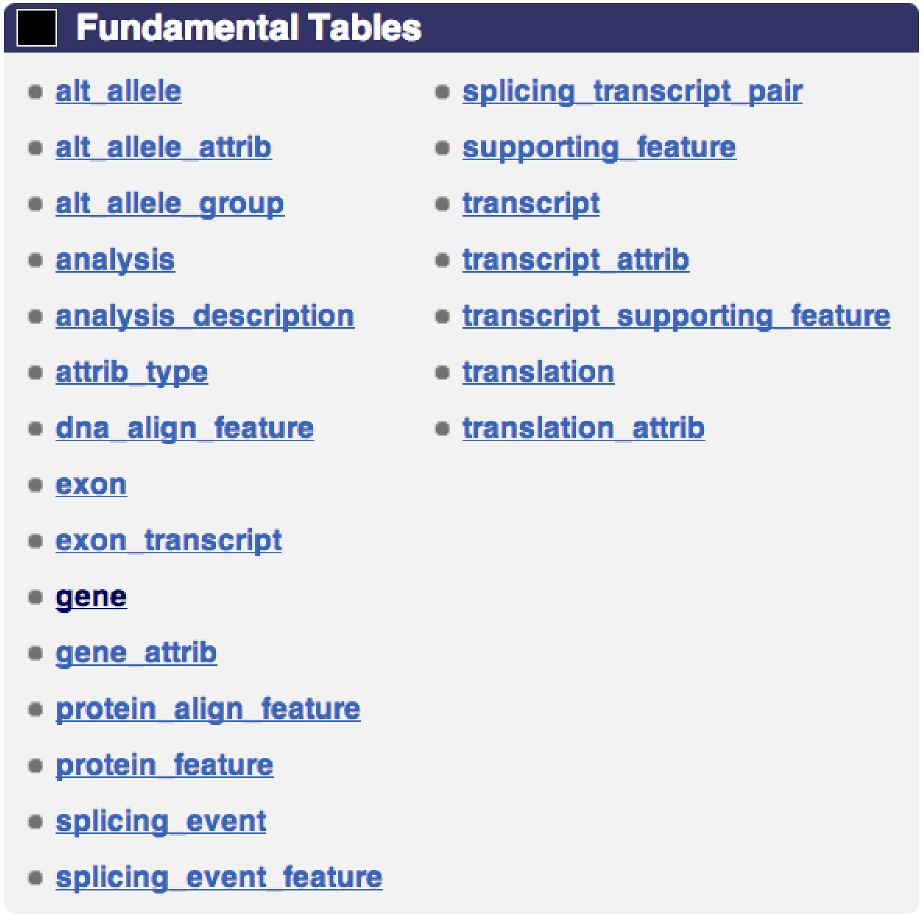
- associated_group
- associated_xref
- dependent_xref
- external_db
- external_synonym

Features

- density_feature
- density_type
- ditag
- ditag_feature
- intron_supporting_evidence
- prediction_transcript
- repeat_consensus
- repeat_feature
- simple_feature
- transcript_intron_supporting_evidence

Ensembl Core DB Documentation (2)

Scroll down to the list of Fundamental Tables and click on the gene table.

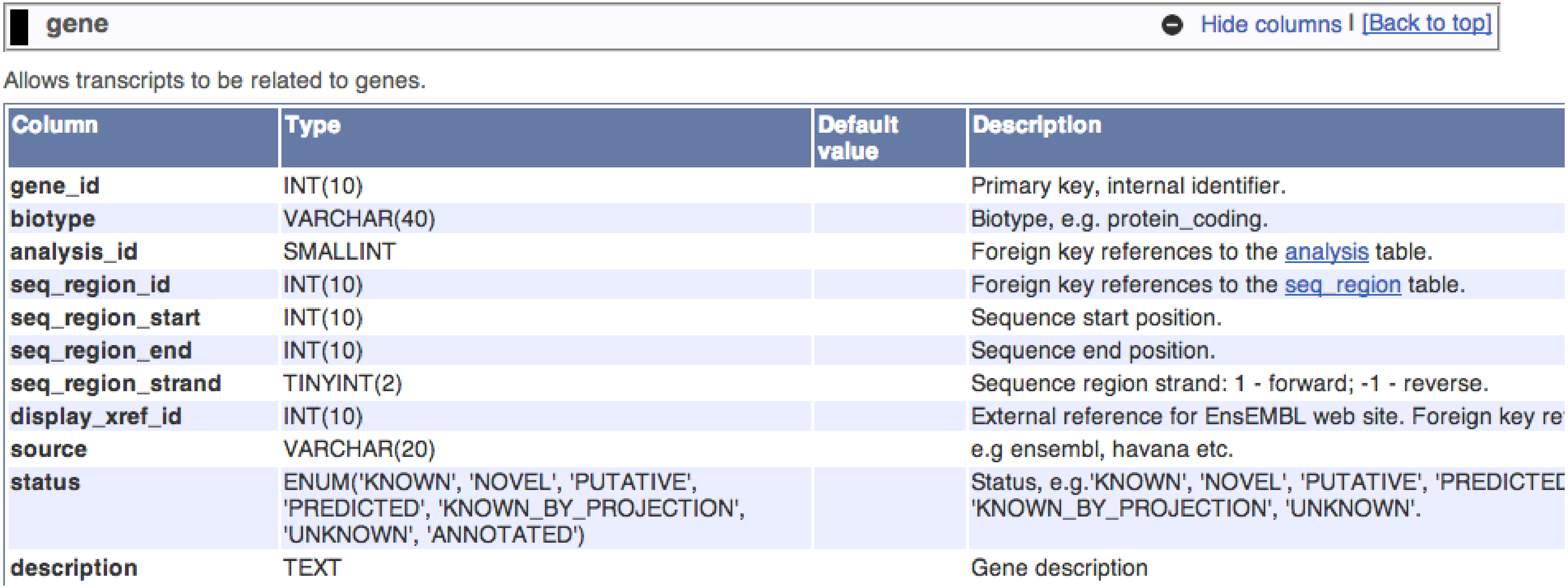


Fundamental Tables

- alt_allele
- alt_allele_attrib
- alt_allele_group
- analysis
- analysis_description
- attrib_type
- dna_align_feature
- exon
- exon_transcript
- gene
- gene_attrib
- protein_align_feature
- protein_feature
- splicing_event
- splicing_event_feature
- splicing_transcript_pair
- supporting_feature
- transcript
- transcript_attrib
- transcript_supporting_feature
- translation
- translation_attrib

Ensembl Core DB Documentation (3)

Click on the ‘show columns’ link to the right to expand a list of gene table columns, their types, descriptions and indices over the columns.



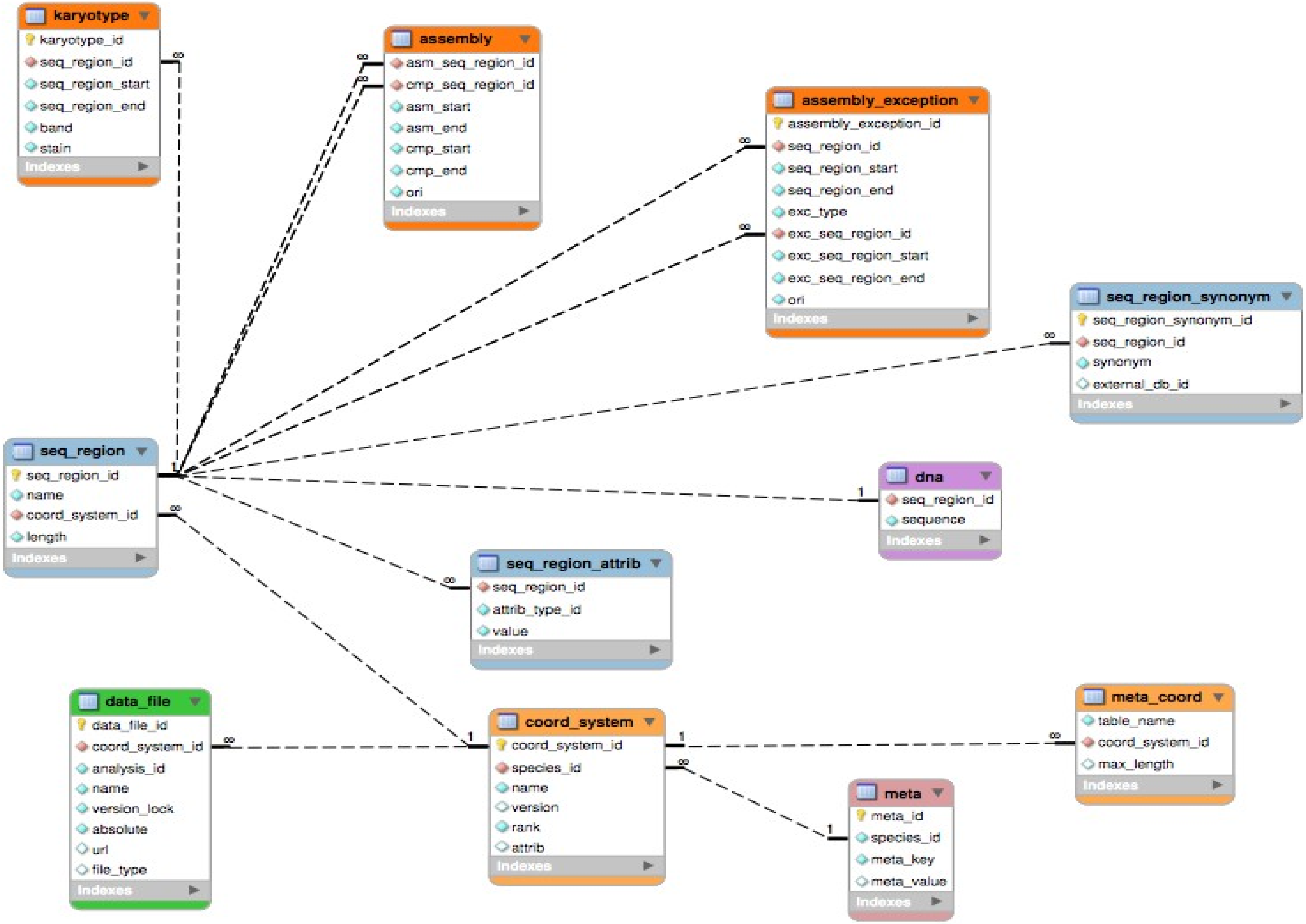
gene

⊖ Hide columns I [Back to top]

Allows transcripts to be related to genes.

Column	Type	Default value	Description
gene_id	INT(10)		Primary key, internal identifier.
biotype	VARCHAR(40)		Biotype, e.g. protein_coding.
analysis_id	SMALLINT		Foreign key references to the analysis table.
seq_region_id	INT(10)		Foreign key references to the seq_region table.
seq_region_start	INT(10)		Sequence start position.
seq_region_end	INT(10)		Sequence end position.
seq_region_strand	TINYINT(2)		Sequence region strand: 1 - forward; -1 - reverse.
display_xref_id	INT(10)		External reference for EnsEMBL web site. Foreign key re
source	VARCHAR(20)		e.g ensembl, havana etc.
status	ENUM('KNOWN', 'NOVEL', 'PUTATIVE', 'PREDICTED', 'KNOWN_BY_PROJECTION', 'UNKNOWN', 'ANNOTATED')		Status, e.g.'KNOWN', 'NOVEL', 'PUTATIVE', 'PREDICTED 'KNOWN_BY_PROJECTION', 'UNKNOWN'.
description	TEXT		Gene description

Ensembl Core DB Documentation (4)



karyotype
- karyotype_id
- seq_region_id
- seq_region_start
- seq_region_end
- band
- stain
- Indexes

assembly
- asm_seq_region_id
- cmp_seq_region_id
- asm_start
- asm_end
- cmp_start
- cmp_end
- ori
- Indexes

assembly_exception
- assembly_exception_id
- seq_region_id
- seq_region_start
- seq_region_end
- exc_type
- exc_seq_region_id
- exc_seq_region_start
- exc_seq_region_end
- ori
- Indexes

seq_region_synonym
- seq_region_synonym_id
- seq_region_id
- synonym
- external_db_id
- Indexes

seq_region
- seq_region_id
- name
- coord_system_id
- length
- Indexes

dna
- seq_region_id
- sequence
- Indexes

seq_region_attrib
- seq_region_id
- attrib_type_id
- value
- Indexes

data_file
- data_file_id
- coord_system_id
- analysis_id
- name
- version_lock
- absolute
- url
- file_type
- Indexes

coord_system
- coord_system_id
- species_id
- name
- version
- rank
- attrib
- Indexes

meta_coord
- table_name
- coord_system_id
- max_length
- Indexes

meta
- meta_id
- species_id
- meta_key
- meta_value
- Indexes

Exercise 1

a) Find documentation for the Exon class in the Ensembl core code base. Which method would you use to retrieve the DNA sequence for an exon? What is the return type for this method?

b) Can you find a table which stores stable ids for transcripts? Which table stores DNA sequence? How many columns does this table have?

Outline

a. Introduction

b. Data objects & object adaptors

c. Ensembl documentation

d. The Registry & Ensembl API script design

e. Coordinate systems & slices

f. Features

g. Genes, transcripts, exons & translations

h. External references

The Registry

We know how to use Data Objects and Object Adaptors.

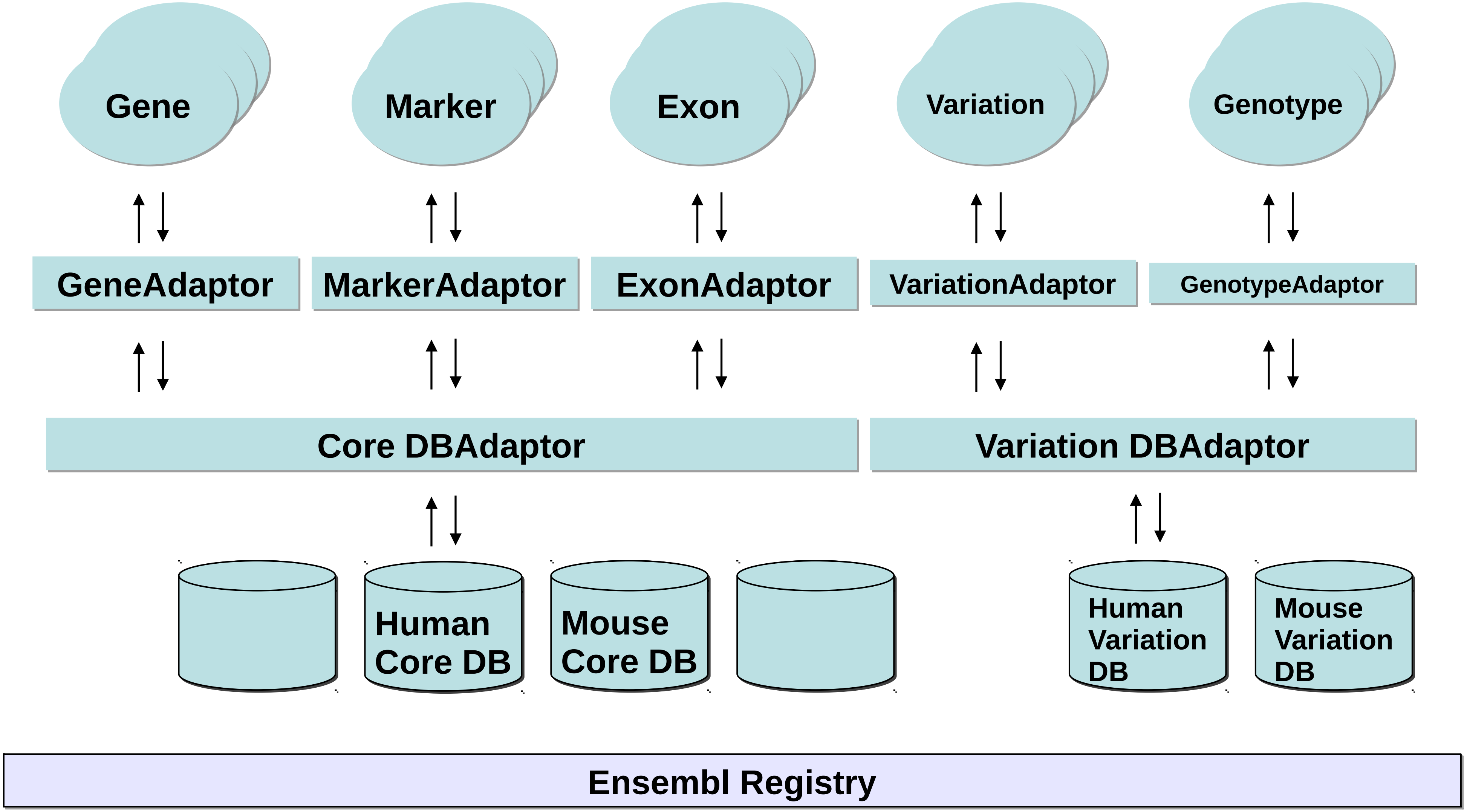
How do we make sure we get those from the right database?

This is what we use the Registry for.

The Registry:

- loads all databases of the same version as the API
- lazy loads so no connections are made until requested

Ensembl API Architecture



Review: The basic Ensembl script

```
use strict;
Use warnings
use Bio::EnsEMBL::Registry;

my $reg = ‘Bio::EnsEMBL::Registry’;

$reg->load_registry_from_db(
    -host => ‘ensembldb.ensembl.org’,
    -user => ‘anonymous’
);

my $ga = $reg->get_adaptor(‘human’,‘core’,‘gene’);


...

my $gene= $ga->fetch_by_stable_id(‘ENSG00000139618’);
```

Exercise 2

Create a script which uses the method load_registry_from_db to load all databases into the Registry and prints the names of the databases loaded.

Hint: Have a look at the Doxygen documentation for the Registry object and method load_registry_from_db (http://www.ensembl.org/info/docs/Doxygen/index.html).

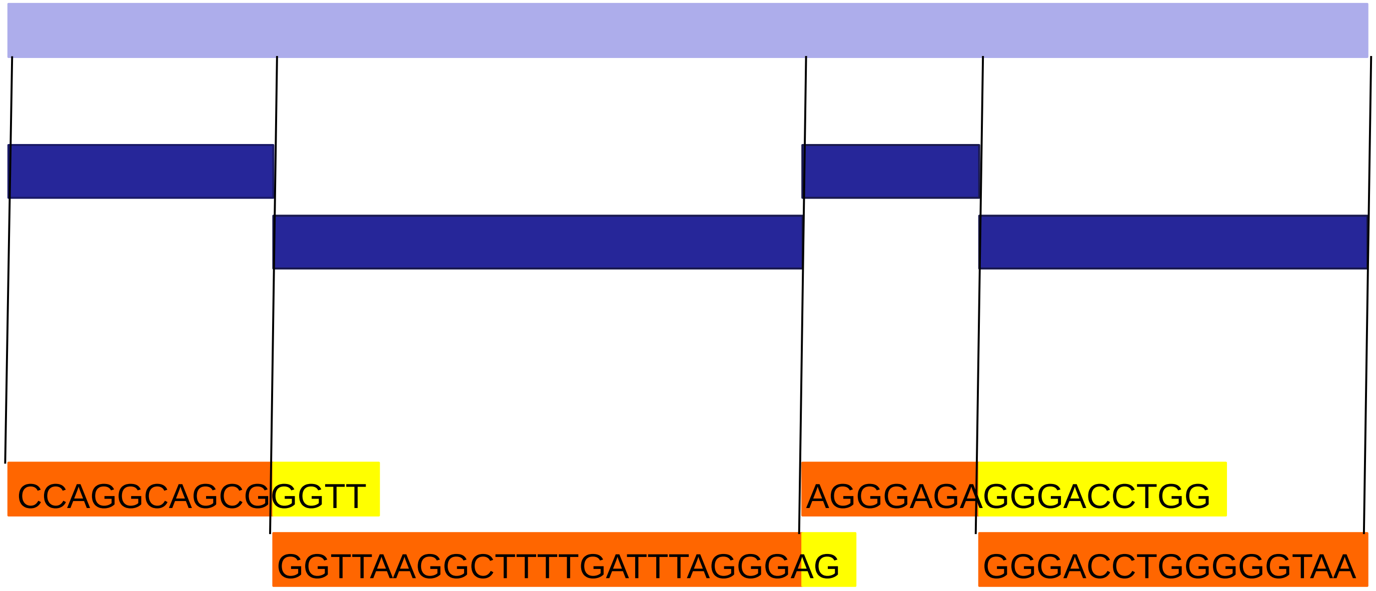
Outline

a. Introduction
b. Data objects & object adaptors
c. Ensembl documentation
d. The Registry & Ensembl API script design
e. Coordinate systems & slices
f. Features
g. Genes, transcripts, exons & translations
h. External references

Coordinate Systems (1)

Ensembl stores features and DNA sequence in a number of coordinate systems.

Top level coordinate system



Chromosome

Contigs

Clones
(Tiling path)

CCAGGCAGCGGGTT

AGGGAGAGGGACCTGG

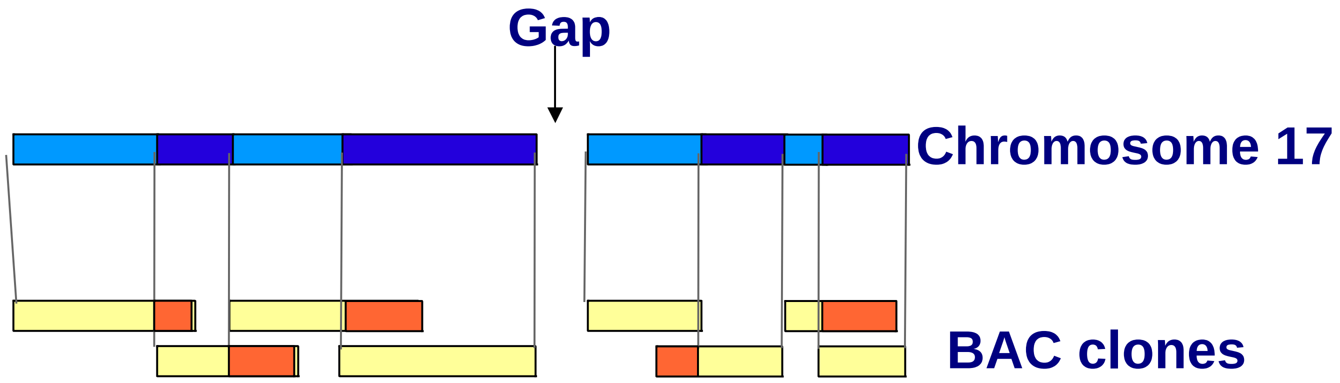
GGTTAAGGCTTTTGATTTAGGGAG

GGGACCTGGGGGTAA

Sequence level coordinate system

Coordinate Systems (2)

Regions in one coordinate system may be constructed from a tiling path of regions from another coordinate system.



Coordinate Systems - Code Example

```
#   Obtain all coordinate systems for human

use Bio::EnsEMBL::Registry;
my $registry = 'Bio::EnsEMBL::Registry';

$registry->load_registry_from_db(
    -host => 'ensembldb.ensembl.org',
    -user => 'anonymous'
);
my $coordsystem_adaptor = $registry->get_adaptor( 'Human', 'Core', 'CoordSystem' );

my $coordsystems = $coordsystem_adaptor->fetch_all;

while ( my $coordsystem = shift @{$coordsystems} ){
    print $coordsystem->name, "\t",
          $coordsystem->version, "\t",
          $coordsystem->rank ,"\n";
}
```

Note use of ‘while’ and ‘shift’ instead of ‘foreach’ – more memory efficient way for large datasets

Coordinate Systems – Code Output

```
OUTPUT:
name                    version     rank

chromosome      GRCh38       1
scaffold        GRCh38       2
clone                        3

contig                       4



chromosome      GRCh37       5
chromosome      NCBI36       6
chromosome      NCBI35       7
chromosome      NCBI34       7


lrg                          8
```

Latest assembly, top level

Latest assembly, sequence level

Old assemblies, used for mapping features between assembly versions.

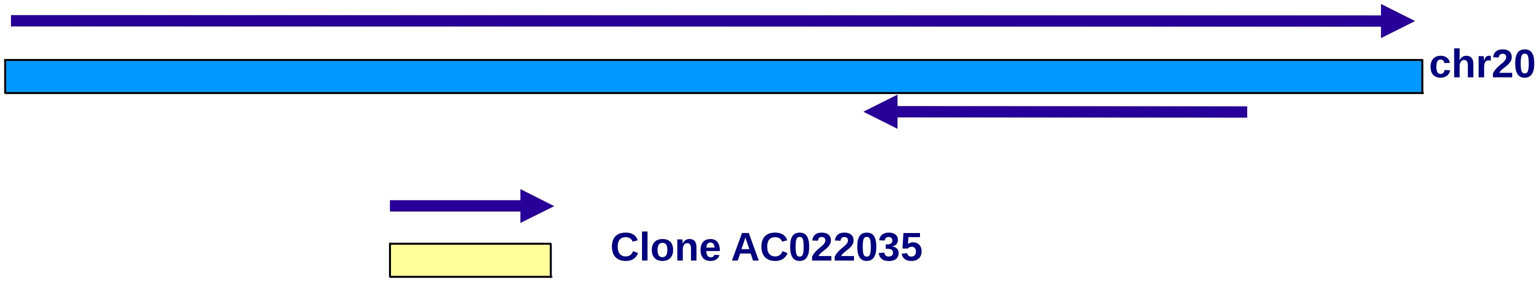
Locus-Reference Genes, used in clinical tests

Slices

A *Slice* Data Object represents an arbitrary region of a genome, a slice of a Sequence Region.

Slices are not directly stored in the database.

A *Slice* is used to request sequence or features from a specific region in a specific coordinate system.



Slices - Code Example (1)

```
# get a slice covering the entire human Y chromosome

my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );
my $slice = $slice_adaptor->fetch_by_region( 'chromosome', 'Y' );
print  "Coord system:\t", $slice->coord_system_name, "\n",
       "Seq region:\t", $slice->seq_region_name, "\n",
       "Start:\t\t", $slice->start,"\n",
       "End:\t\t", $slice->end,"\n",
       "Strand:\t\t", $slice->strand,"\n",
       "Slice:\t\t", $slice->name, "\n";
```

OUTPUT:

```
Coord system:   chromosome
Seq region: Y
Start:      1
End:        57227415
Strand:     1
Slice:      chromosome:GRCh38:Y:1:57227415:1
```

Slices - Code Example (2)

```
# get the slice adaptor
my $slice_adaptor = $registry->get_adaptor('human', 'core', 'slice');

# fetch a slice on a region of chromosome 12
my $slice = $slice_adaptor->fetch_by_region('chromosome', '12',
                                          1e6, 2e6);


# print out the sequence from this region
print $slice->seq, "\n";

# get all clones in the database and print out their names
my @slices = @{ $slice_adaptor->fetch_all('clone') };
while ( my $slice = shift @{$slices} ){
print $slice->seq_region_name, "\n";
}
```

Exercise 3

(a) Fetch all chromosomes for human. Determine their number and print the name and length for each of them. *The number of chromosomes is probably not what you would expect! Why is this?*

(b) Use the gene stable id 'ENSG00000101266' to fetch a slice surrounding this gene with 2kb of flanking sequence.

(a)hint: use the Ensembl API documentation to find an appropriate method in SliceAdaptor class which retrieves a slice given a gene stable id (pay attention to the method’s arguments)

(c) Fetch the sequence of the first 10Mb of chromosome 20 and write it to a file in FASTA format. Print the number of genes in this region.

✓ hint: Slice objects inherit from Bio::Seq so can be written to file easily using Bio::SeqIO, e.g.:

```
my $output = Bio::SeqIO->new( -file=>'>filename.fasta',
   -format=>'Fasta');
$output->write_seq($slice);
```

Outline

a. Introduction

b. Data objects & object adaptors

c. Ensembl documentation

d. The Registry & Ensembl API script design

e. Coordinate systems & slices

f. Features

g. Genes, transcripts, exons & translations

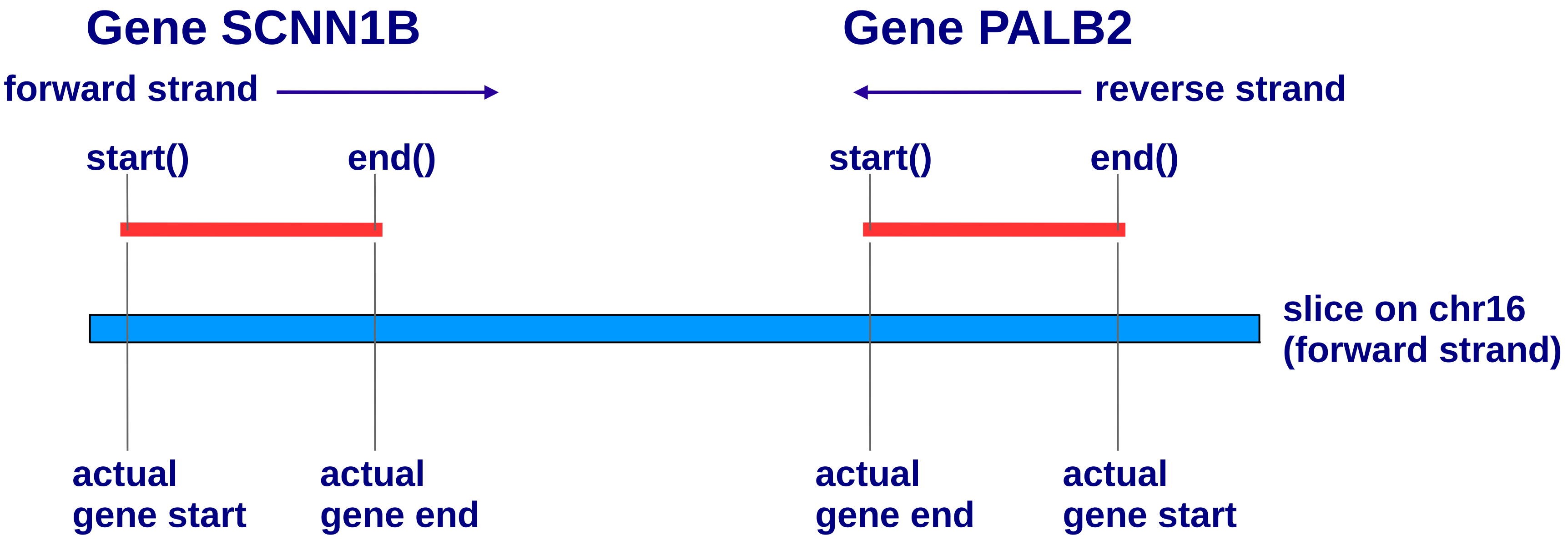
h. External references

Features (1)

Features have a defined location on the genome and are stored in a single coordinate system

All *Features* have a *start*, *end*, *strand* and *slice*

Start and *end* are plotted onto the forward strand: start < end

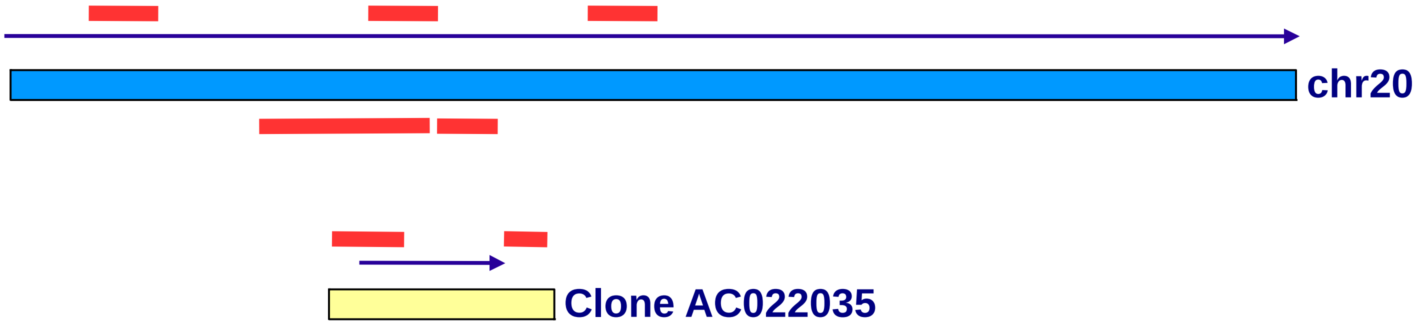


Features (2)

slice() method returns the Slice object with which the *Feature* is associated

feature_Slice() method returns the *Slice* object which covers only the *Feature* (will start at 1 and end with *Feature* length)

Features are retrieved from Object Adaptors using identifiers or regions (slices).



Features Objects – Biological Correspondence

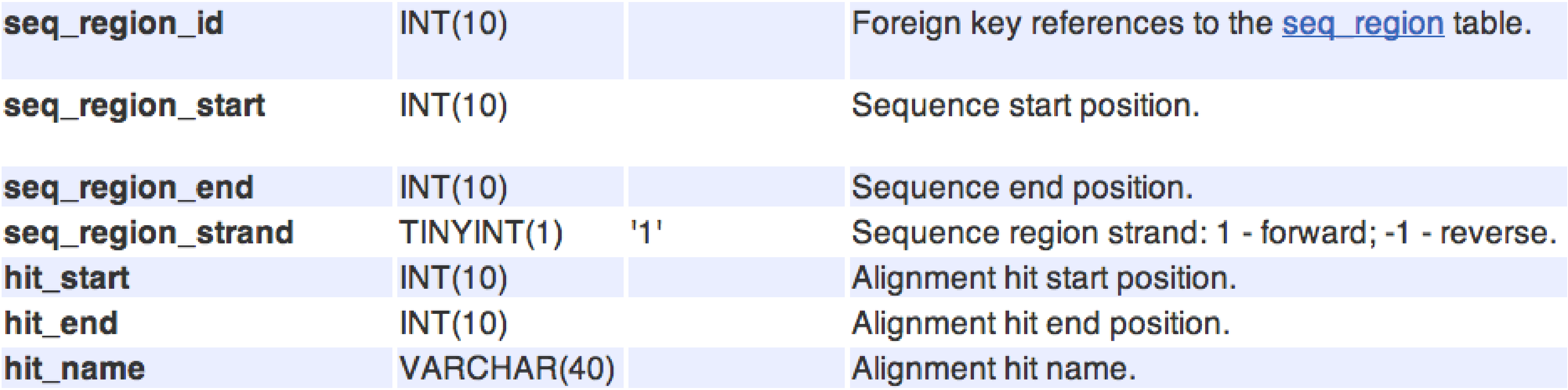
Object	Biological entity
Gene, Transcript, Exon	**Ensembl gene models**
PredictionTranscript, PredictionExon	Genscan gene models
DNAAlignFeature, ProteinAlignFeature	cDNAs, proteins
RepeatFeature	**repeats**
MarkerFeature	markers
KaryotypeBand	cytogenetic bands
SimpleFeature	results of cpg, Eponine, FirstEF and tRNAscan
MiscFeature	clones, ENCODE regions
ProteinFeature	protein domains

Align features

A *sequence* (mRNA or protein) is aligned against the *genome*

The result is stored in an *align_feature* table (*dna_align_feature* or *protein_align_feature*)

A row represents the alignment between the genome sequence (a slice) and the target sequence (a hit)



seq_region_id	INT(10)		Foreign key references to the seq_region table.
seq_region_start	INT(10)		Sequence start position.
seq_region_end	INT(10)		Sequence end position.
seq_region_strand	TINYINT(1)	'1'	Sequence region strand: 1 - forward; -1 - reverse.
hit_start	INT(10)		Alignment hit start position.
hit_end	INT(10)		Alignment hit end position.
hit_name	VARCHAR(40)		Alignment hit name.

Exercise 4

(a) Get all the repeat features from chromosome 20:1-500kb. Print out the name and position of each on the chromosome and the total number.

- – hint: create a slice, retrieve repeat features on this slice

(b) Find which genomic region the RefSeq dna entry NM_000059.3 was mapped to. Print the name of the region and coordinates of the alignment on the genome as well as the name of the region and coordinates of the alignment on the RefSeq dna entry. Print the score and percentage identity for the alignment.

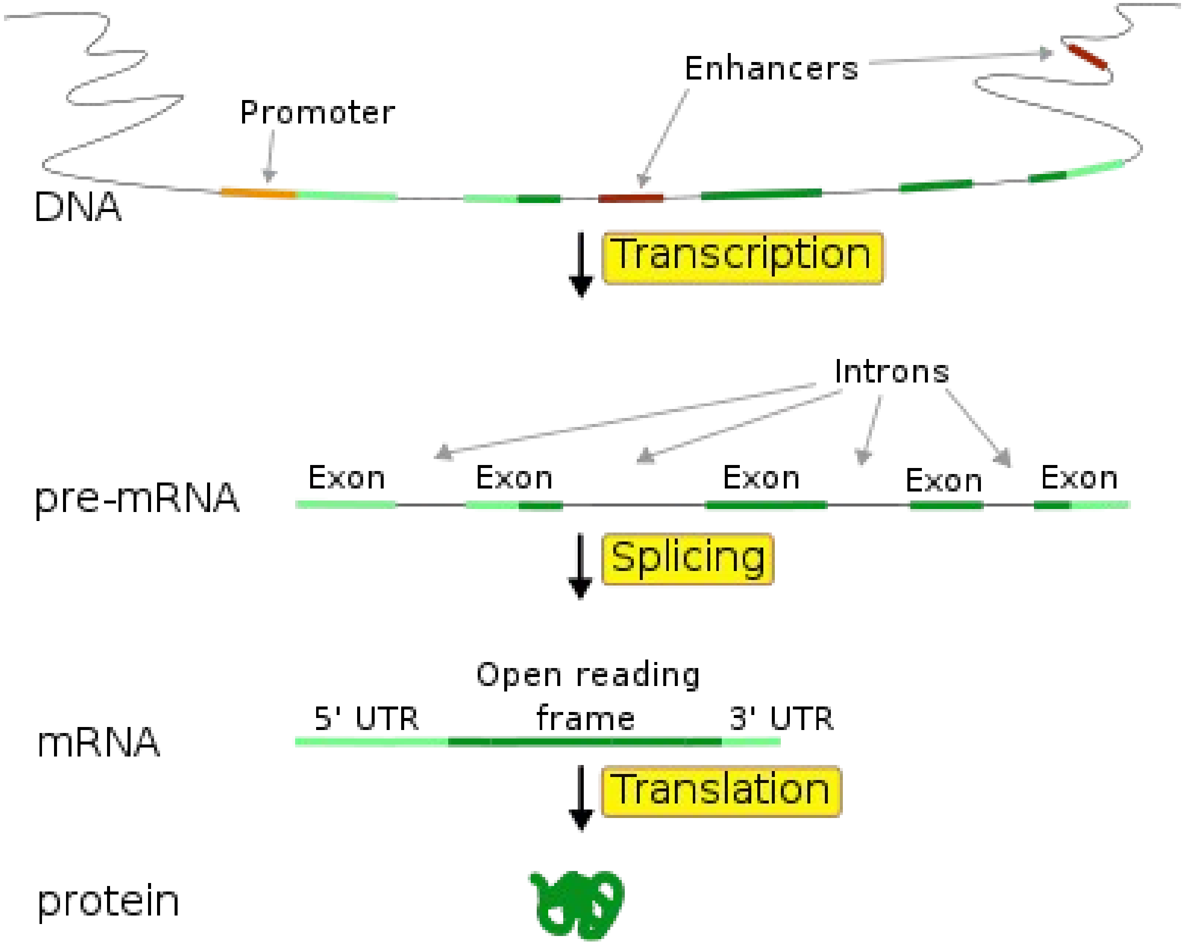
- ✓ hint: use DnaAlignFeatureAdaptor; use the core schema documentation as a guide to appropriate methods which correspond to columns in dna_align_feature table

A list of useful Feature methods is in the Appendix at the end of the presentation slides

Outline

a. Introduction
b. Data objects & object adaptors
c. Ensembl documentation
d. The Registry & Ensembl API script design
e. Coordinate systems & slices
f. Features
g. Genes, transcripts, exons & translations
h. External references

Genes, transcripts, exons & translations



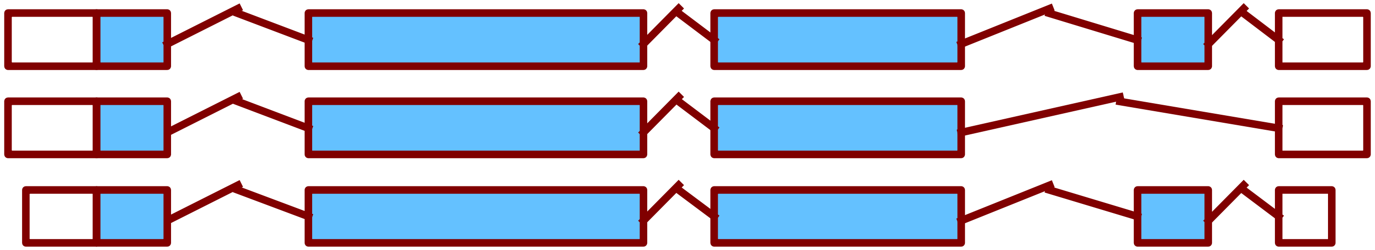
Genes, Transcripts and Exons

Genes, Transcript and Exons are objects that can be used just like any other Feature object

A Gene is a set of alternatively spliced Transcripts

A Transcript is a set of Exons

Introns are not explicitly defined in the database



Genes, Transcripts, Exons – Code Output

```
# helper function: returns location and stable_id string for a feature
sub get_string {
  my $feature = shift;
  my $stable_id  = $feature->stable_id;
  my $seq_region = $feature->slice->seq_region_name;
  my $start = $feature->start;
  my $end = $feature->end;
  my $strand = $feature->strand;
  return "$stable_id  $seq_region:$start-$end($strand)";
}
# fetch a gene by its stable identifier
my $gene = $gene_adaptor->fetch_by_stable_id('ENSG00000123427');

# print out the gene, its transcripts, and its exons
print "Gene: ", get_string($gene), "\n";
while ( my $transcript = shift @{$gene->get_all_Transcripts} ){
  print "  Transcript: ", get_string($transcript), "\n";
  while ( my $exon = shift @{$transcript->get_all_Exons} ){
    print "    Exon: ", get_string($exon), "\n";
  }
}
```

Genes, Transcripts, Exons – Code Output

```
OUTPUT:
Gene: ENSG00000123427  12:57771492-57782541(1)
  Transcript: ENST00000548256  12:57771492-57780430(1)
    Exon: ENSE00002360002  12:57771492-57771625(1)
    Exon: ENSE00003631087  12:57773017-57773128(1)
    Exon: ENSE00003530124  12:57774629-57774767(1)
    Exon: ENSE00002406112  12:57780255-57780430(1)
  Transcript: ENST00000551420  12:57772200-57780780(1)
    Exon: ENSE00002355737  12:57772200-57772397(1)
    Exon: ENSE00003506271  12:57773017-57773128(1)
    Exon: ENSE00002376752  12:57780255-57780780(1)
  Transcript: ENST00000300209  12:57772600-57782403(1)
    Exon: ENSE00002301479  12:57772600-57772901(1)
    Exon: ENSE00003631087  12:57773017-57773128(1)
    Exon: ENSE00002393444  12:57780255-57782403(1)
                                                 etc.
```

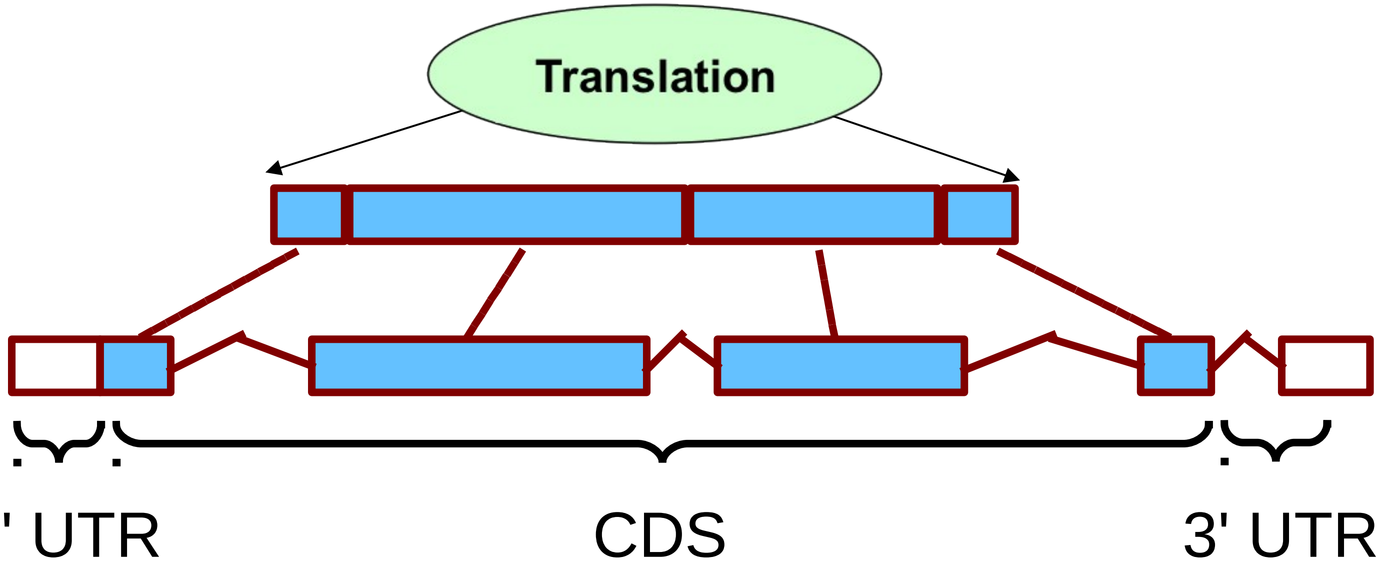
Translations

Translations are not Features.

A Translation object defines the UTR and CDS of a Transcript.

Peptides are not stored in the database, they are computed on the fly using Transcript objects.

Not all transcripts have a translation (e.g. ncRNAs)



Translations – Code Example

```
my $transcript_adaptor = $registry->get_adaptor( 'Homo sapiens', 'Core',
   'Transcript' );
# fetch a transcript from the database
my $transcript =
  $transcript_adaptor->fetch_by_stable_id('ENST00000333012');

# obtain the translation of the transcript
my $translation = $transcript->translation;

# print out the translation info
print "Translation: ", $translation->stable_id, "\n";
print "Start Exon: ",$translation->start_Exon->stable_id,"\n";
print "End Exon:   ", $translation->end_Exon->stable_id, "\n";

# cDNA start and end (spliced sequence with UTR)
print "Start : ", $translation->cdna_start, "\n";
print "End   : ", $translation->cdna_end, "\n";

# print the peptide which is the product of the translation
print "Peptide : ", $transcript->translate->seq, "\n";
```

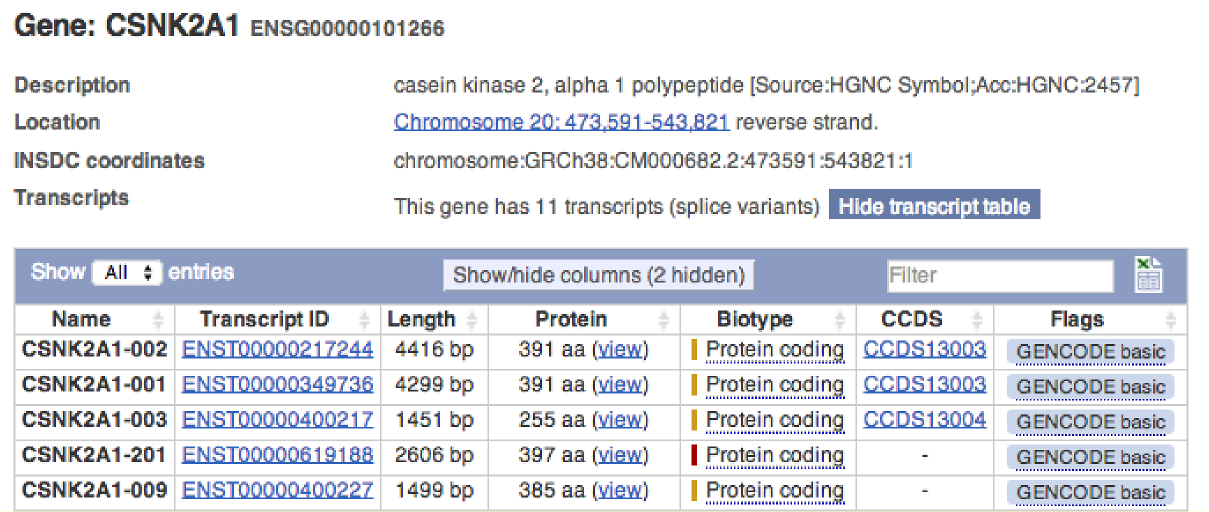
Translations – Code Output

```
OUTPUT:

Translation: ENSP00000327425
Start Exon: ENSE00002340145
End Exon:   ENSE00002428887
Start : 48
End   : 497
Peptide :
   MADPGPDPESESESVFPREVGLFADSYSEKSQFCFCGHVLTITQNFGSRLGVAARVWDAALSLCNYFESQNVDFR
   GKKVIELGAGTGIVGILAALQGAYGLVRETEDDVIEQELWRGMRGACGHALSMSTMTPWESIKGSSVRGGCYHH
```

Exercise 5

(a) Fetch gene ‘CSNK2A1’and print the number of its transcripts and exons.

- ✓ hint: use GeneAdaptor method fetch_by_display_label; remember that not all transcripts have a translation



Gene: CSNK2A1 ENSG00000101266

Description	casein kinase 2, alpha 1 polypeptide [Source:HGNC Symbol;Acc:HGNC:2457]
Location	Chromosome 20: 473,591-543,821 reverse strand.
INSDC coordinates	chromosome:GRCh38:CM000682.2:473591:543821:1
Transcripts	This gene has 11 transcripts (splice variants) Hide transcript table

Name	Transcript ID	Length	Protein	Biotype	CCDS	Flags
CSNK2A1-002	ENST00000217244	4416 bp	391 aa (view)	Protein coding	CCDS13003	GENCODE basic
CSNK2A1-001	ENST00000349736	4299 bp	391 aa (view)	Protein coding	CCDS13003	GENCODE basic
CSNK2A1-003	ENST00000400217	1451 bp	255 aa (view)	Protein coding	CCDS13004	GENCODE basic
CSNK2A1-201	ENST00000619188	2606 bp	397 aa (view)	Protein coding	-	GENCODE basic
CSNK2A1-009	ENST00000400227	1499 bp	385 aa (view)	Protein coding	-	GENCODE basic

(b) For the above gene, get all the transcripts and list the number of exons for each. Also show any translations associated with the transcripts.

(c) Why do the exon numbers not match?

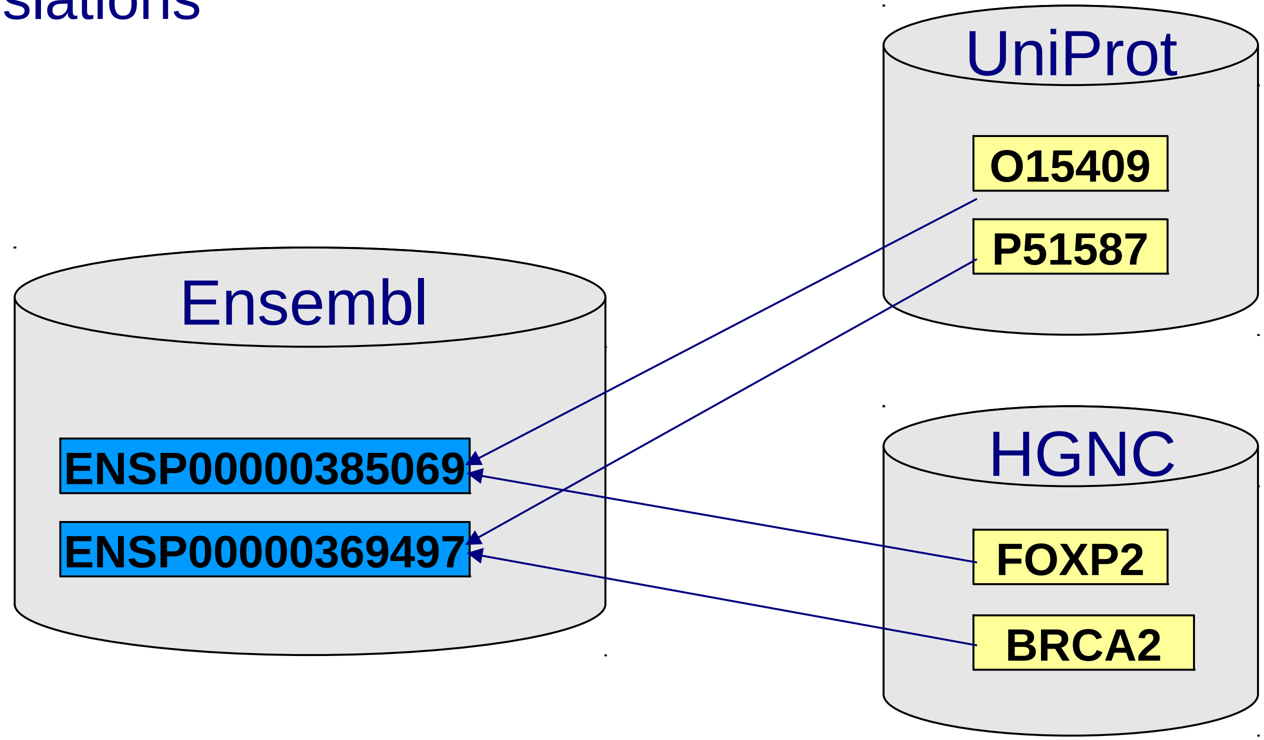
Outline

a. Introduction
b. Data objects & object adaptors
c. Ensembl documentation
d. The Registry & Ensembl API script design
e. Coordinate systems & slices
f. Features
g. Genes, transcripts, exons & translations
h. External references

External References

Ensembl cross references its Gene models with identifiers from other databases, such as HGNC, WikiGenes, UniProtKB/Swiss-Prot, RefSeq, MIM etc.

External References (Xrefs) can be linked to genes, transcripts or translations



Xrefs - Code Example (1)

```
# Obtain external references for Ensembl gene ENSG00000139618

my $gene = $gene_adaptor->fetch_by_stable_id( 'ENSG00000139618' );

my $gene_xrefs = $gene->get_all_DBEntries;
print "Xrefs on gene level: \n\n";

while ( my $gene_xref = shift @{$gene_xrefs} ){
  print $gene_xref->dbname, ":", $gene_xref->display_id, "\n";
}

my $all_xrefs = $gene->get_all_DBLinks;

print "\nXrefs on gene, transcript and protein level: \n\n";
while ( my $all_xref = shift @{$all_xrefs} ){
  print $all_xref->dbname, ":", $all_xref->display_id, "\n";
}
```

this method will only return xrefs linked to the object it’s called on (e.g. gene)

this method will return xrefs on all levels (gene, transcript and translation)

Xrefs – Code Output (1)

OUTPUT:

```
Xrefs on gene level:

Vega_gene:OTTHUMG00000017411
Vega_gene:BRCA2
PUBMED:15057823
PUBMED:7581463
PUBMED:8091231
RefSeq_dna:NM_000059
OTTG:OTTHUMG00000017411
ENS_LRG_gene:LRG_293
ArrayExpress:ENSG00000139618
DBASS3:BRCA2
DBASS5:BRCA2
EntrezGene:BRCA2
HGNC:BRCA2
MIM_GENE: BRCA2 GENE [*600185]
MIM_MORBID: BREAST CANCER [#114480]
MIM_MORBID: GLIOMA SUSCEPTIBILITY 3 [#613029]
MIM_MORBID: PANCREATIC CANCER, SUSCEPTIBILITY  [#613347]
UniGene:Hs.34012
UniGene:Hs.686439
Uniprot_gn:BRCA2
WikiGene:BRCA2

Xrefs on gene, transcript and translation level:
(same as on gene level + transcript and translation
    level)

Vega_gene:OTTHUMG00000017411
Vega_gene:BRCA2
PUBMED:15057823
PUBMED:7581463
PUBMED:8091231
RefSeq_dna:NM_000059
OTTG:OTTHUMG00000017411
ENS_LRG_gene:LRG_293
ArrayExpress:ENSG00000139618
DBASS3:BRCA2
DBASS5:BRCA2
EntrezGene:BRCA2
HGNC:BRCA2
Uniprot/SPTREMBL:E9PIQ1
Uniprot/SPTREMBL:K4JTT2

    etc.
```

Xrefs - Code Example (2)

```
# Retrieve Ensembl IDs for a list of UniProt protein IDs

my $translation_adaptor
  = $registry->get_adaptor(
    'Human','Core','Translation');


my @uniprot_ids = qw(P51587 P15056 B8A597 B8A595 B7ZW72);

while ( my $uniprot_id = shift @{$uniprot_ids} ){
  my @trans = @{
   $translation_adaptor->fetch_all_by_external_name($uniprot_id,'Uniprot%')
  };

  while ( my $translation = shift @{$trans} ){
    print $translation->stable_id."\t".$uniprot_id."\n";
  }
}
```

Proteins map to Ensembl Translation objects so we will use a TranslationAdaptor

Xrefs – Code Output (2)

OUTPUT:

```
ENSP00000439902
   P51587
ENSP00000369497
   P51587
ENSP00000288602
   P15056
ENSP00000387217
   B8A597
ENSP00000386781
   B8A595
ENSP00000307640
   B7ZW72
```

Cross references can map to more than one Ensembl identifier

Exercise 6

Retrieve a list of GO term IDs and term names linked to the gene with stable id ‘ENSG00000139618’

- ✓ Use *get_all_DBLinks* with an external database name argument to restrict the number of xrefs returned
- ✓ Ontology term data such as term name and definition are stored outside of the core database. Create an OntologyTerm Adaptor with the help of the Registry method *get_adaptor* using arguments: ‘Multi’ (species), ‘Ontology’ (database type), ‘OntologyTerm’ (adaptor type)
- ✓ For all xrefs returned by *get_all_DBLinks* use the OntologyTerm Adaptor to fetch the relevant term and print its accession and name (xref display_id is the same as term accession)

Recap - Ensembl API script design

Always:

- Load the registry

Which features (genes, repeats, SNPs, etc.) are in my particular region of interest?

- Get the SliceAdaptor
- Fetch the Slice for you region of interest
- Get the features from your Slice

What do we know about a particular gene (or any other feature)?

- Get the GeneAdaptor
- Fetch your Gene of interest
- Get more details about the gene:
  - Gene structure (transcripts, exons, translations)
  - Annotations: GO xrefs, HGNC symbols, etc.
  - Features in the same region -> get the Slice for the Gene!

Documentation & Help

- Installation instructions, web-browsable version of the POD (Perldoc), database schema and tutorial:

  http://www.ensembl.org/info/docs/api/index.html

- Inline Perl POD (Plain Old Documentation)

- dev@ensembl.org mailing list:

  http://www.ensembl.org/info/about/contact/mailing.html

  searchable mailing list archive:

  http://blog.gmane.org/gmane.science.biology.ensembl.devel

- Ensembl helpdesk:

  helpdesk@ensembl.org

Ensembl Acknowledgements

The Entire Ensembl Team

Bronwen L. Aken[1], Premanand Achuthan[1], Wasiu Akanni[1], M. Ridwan Amode[1], Friederike Bernsdorff[1], Jyothish Bhai[1], Konstantinos Billis[1], Denise Carvalho-Silva[1], Carla Cummins[1], Peter Clapham[2], Laurent Gil[1], Carlos García Girón[1], Leo Gordon[1], Thibaut Hourlier[1], Sarah E. Hunt[1], Sophie H. Janacek[1], Thomas Juettemann[1], Stephen Keenan[1], Matthew R. Laird[1], Ilias Lavidas[1], Thomas Maurel[1], William McLaren[1], Benjamin Moore[1], Daniel N. Murphy[1], Rishi Nag[1], Victoria Newman[1], Michael Nuhn[1], Chuang Kee Ong[1], Anne Parker[1], Mateus Patricio[1], Harpreet Singh Riat[1], Daniel Sheppard[1], Helen Sparrow[1], Kieron Taylor[1], Anja Thormann[1], Alessandro Vullo[1], Brandon Walts[1], Steven P. Wilder[1], Amonida Zadissa[1], Myrto Kostadima[1], Fergal J. Martin[1], Matthieu Muffato[1], Emily Perry[1], Magali Ruffier[1], Daniel M. Staines[1], Stephen J. Trevanion[1], Fiona Cunningham[1], Andrew Yates[1], Daniel R. Zerbino[1] and Paul Flicek[1,2,*]

[1]European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Genome Campus, Hinxton, Cambridge CB10 1SD, UK and [2]Wellcome Trust Sanger Institute, Wellcome Genome Campus, Hinxton, Cambridge, CB10 1SA, UK

Funding

wellcome

EMBL

BBSRC bioscience for the future

National Human Genome Research Institute

TRANSFORMING GENETIC MEDICINE INITIATIVE

Open Targets

MedBioinformatics

Co-funded by the European Union

Ensembl 2016

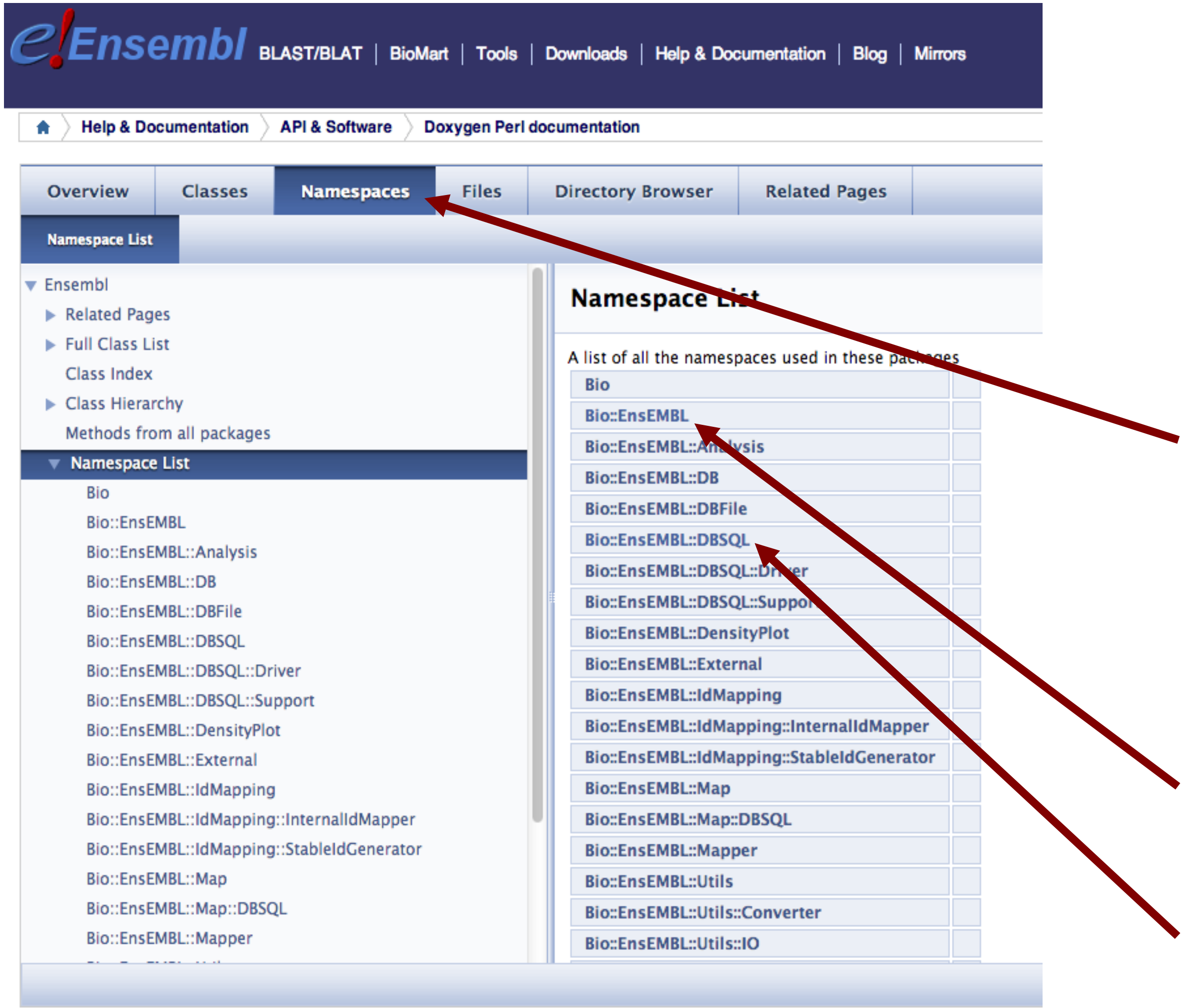
Feedback

Please give us your feedback on the course!

http://training.ensembl.org/events/2017/2017-02-09-APITaiwan

(https://goo.gl/ozhBs8)

Appendix – Adaptors



- Ensembl
  - Related Pages
  - Full Class List
  - Class Index
  - Class Hierarchy
  - Methods from all packages
  - Namespace List
    - Bio
    - Bio::EnsEMBL
    - Bio::EnsEMBL::Analysis
    - Bio::EnsEMBL::DB
    - Bio::EnsEMBL::DBFile
    - Bio::EnsEMBL::DBSQL
    - Bio::EnsEMBL::DBSQL::Driver
    - Bio::EnsEMBL::DBSQL::Support
    - Bio::EnsEMBL::DensityPlot
    - Bio::EnsEMBL::External
    - Bio::EnsEMBL::IdMapping
    - Bio::EnsEMBL::IdMapping::InternalIdMapper
    - Bio::EnsEMBL::IdMapping::StableIdGenerator
    - Bio::EnsEMBL::Map
    - Bio::EnsEMBL::Map::DBSQL
    - Bio::EnsEMBL::Mapper

Namespace List

A list of all the namespaces used in these packages

Bio
Bio::EnsEMBL
Bio::EnsEMBL::Analysis
Bio::EnsEMBL::DB
Bio::EnsEMBL::DBFile
Bio::EnsEMBL::DBSQL
Bio::EnsEMBL::DBSQL::Driver
Bio::EnsEMBL::DBSQL::Support
Bio::EnsEMBL::DensityPlot
Bio::EnsEMBL::External
Bio::EnsEMBL::IdMapping
Bio::EnsEMBL::IdMapping::InternalIdMapper
Bio::EnsEMBL::IdMapping::StableIdGenerator
Bio::EnsEMBL::Map
Bio::EnsEMBL::Map::DBSQL
Bio::EnsEMBL::Mapper
Bio::EnsEMBL::Utils
Bio::EnsEMBL::Utils::Converter
Bio::EnsEMBL::Utils::IO

"What kinds of adaptors are there?"

The online API documentation lists them all under their respective name spaces.

Bio::EnsEMBL for Data Objects

Bio::EnsEMBL::DBSQL for Adaptors

Appendix – CoordSystem Methods

Attribute	Example value(s)	Method(s)
name	`chromosome, scaffold, contig, clone`	`$coordsystem->name`
version	`GRCh37, NCBI36, NCBIM37`	`$coordsystem->version`

Appendix – Feature Methods

Attribute	Example value(s)	Method(s)
name	AluSp, D1S2217	$feature->display_id
coordinates		$feature->seq_region_name $feature->start $feature->end $feature->seq_region_start $feature->seq_region_end $feature->strand
sequence		$feature->seq
length	399	$feature->length

slice relative

chromosome relative

Appendix – Gene Methods

Attribute	Example value(s)	Method(s)
stable ID	`ENSG00000139618`	`$gene->stable_id`
name	`BRCA2`	`$gene->external_name`
description	`breast cancer 2, early onset`	`$gene->description`
biotype	`protein_coding, miRNA`	`$gene->biotype`
analysis	`ensembl, havana, ensembl_havana_gene`	`$gene->analysis->logic_name`

Appendix - Transcript Methods

Attribute	Example value(s)	Method(s)
stable ID	`ENST00000380152`	`$transcript->stable_id`
name	`BRCA2-001`	`$transcript->external_name`
biotype	`protein_coding, nonsense_mediated_decay`	`$transcript->biotype`
analysis	`ensembl, havana, ensembl_havana_transcript`	`$transcript->analysis->logic_name`
status	`KNOWN, NOVEL`	`$transcript->status`
	`ATGCCTATTGGATCCAAAGAGAGGC...`	`$transcript->translateable_seq`

Appendix - Translation Methods

Attribute	Example value(s)	Method(s)
stable id	ENSP00000369497	$translation->stable_id
length	3418	$translation->length
sequence	MPIGSKERPTFFEIFKTRCNKADLG...	$translation->seq