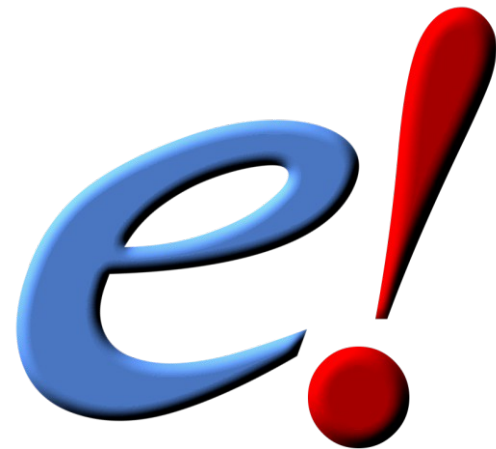
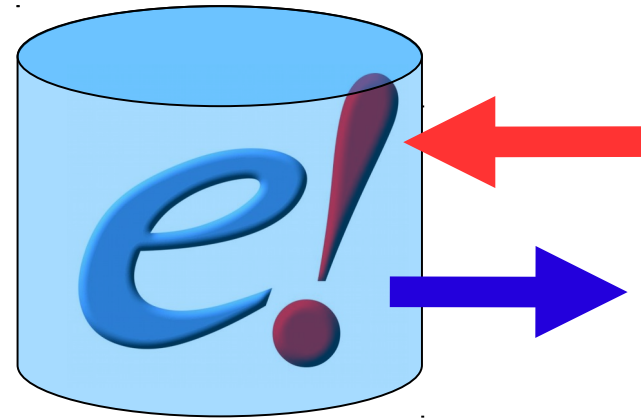


# Ensembl Core API



# 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>

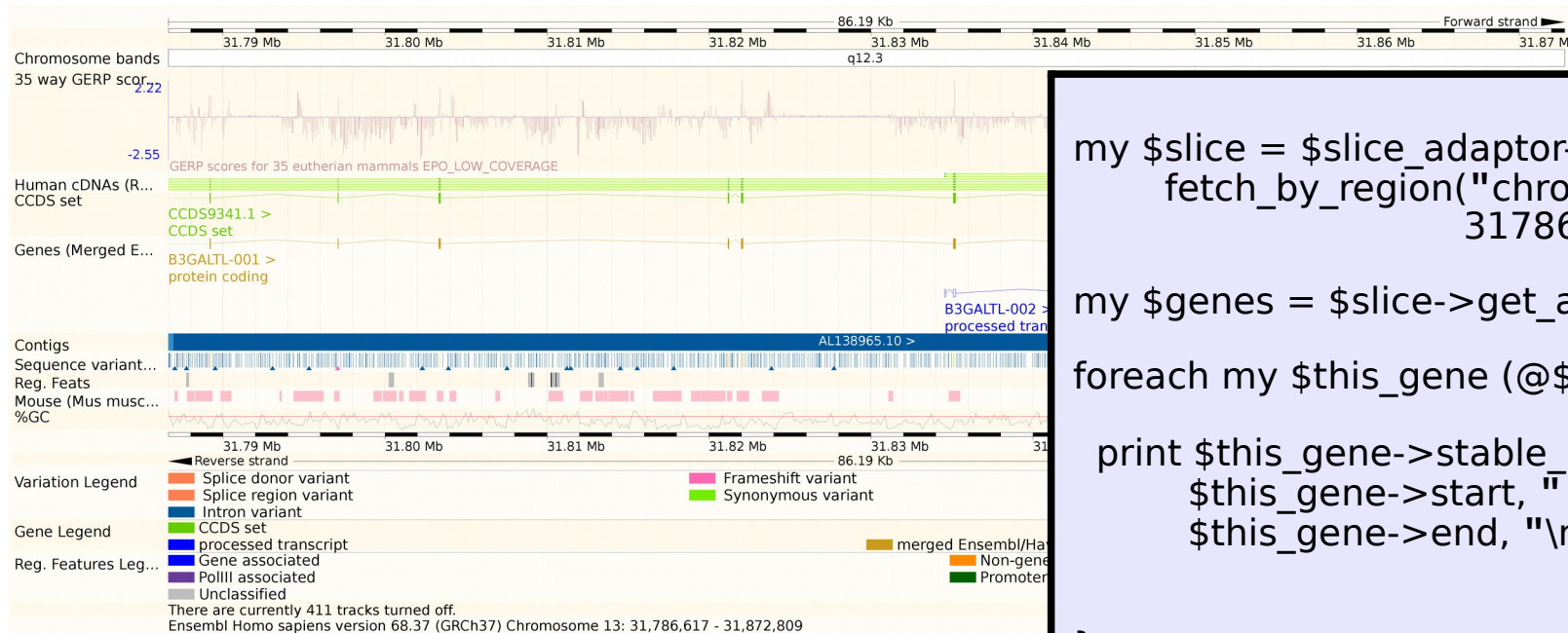
# Outline

- 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.



```
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";

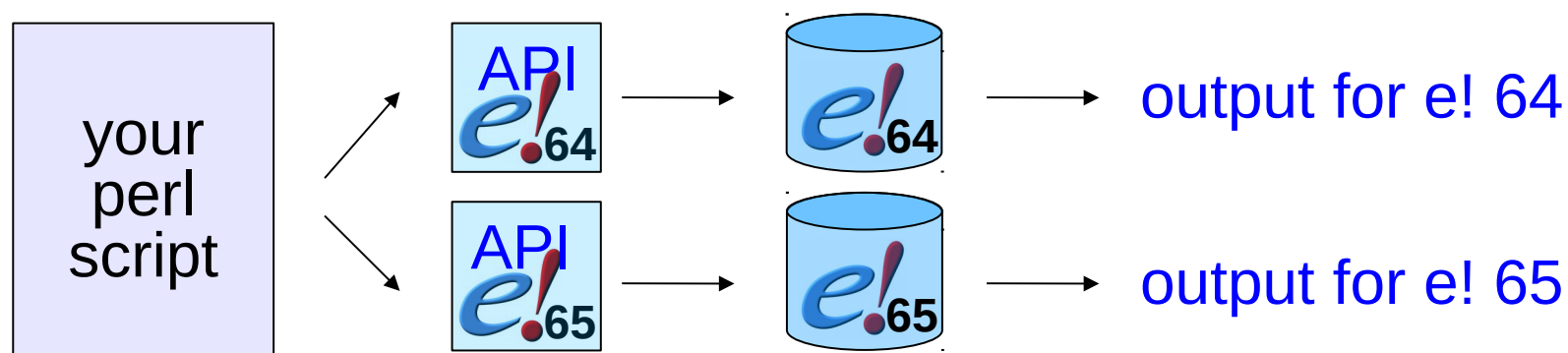
}
```

# 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

[Endpoints](#)[User Guide](#)[Change Log](#)[About the Ensembl Project](#)[Contact Ensembl](#)

## Sequences

Resource	Description
<a href="#">GET sequence/id/:id</a>	Request multiple types of sequence by stable identifier.
<a href="#">GET sequence/region/:species/:region</a>	Returns the genomic sequence of the specified region of the given species.

## Variation

Resource	Description
<a href="#">GET variation/:species/:id</a>	Uses a variation identifier (e.g. rsID) to return the variation features
<a href="#">GET vep/:species/id/:id</a>	Fetch variant consequences based on a variation identifier
<a href="#">POST vep/:species/id/</a>	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](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.

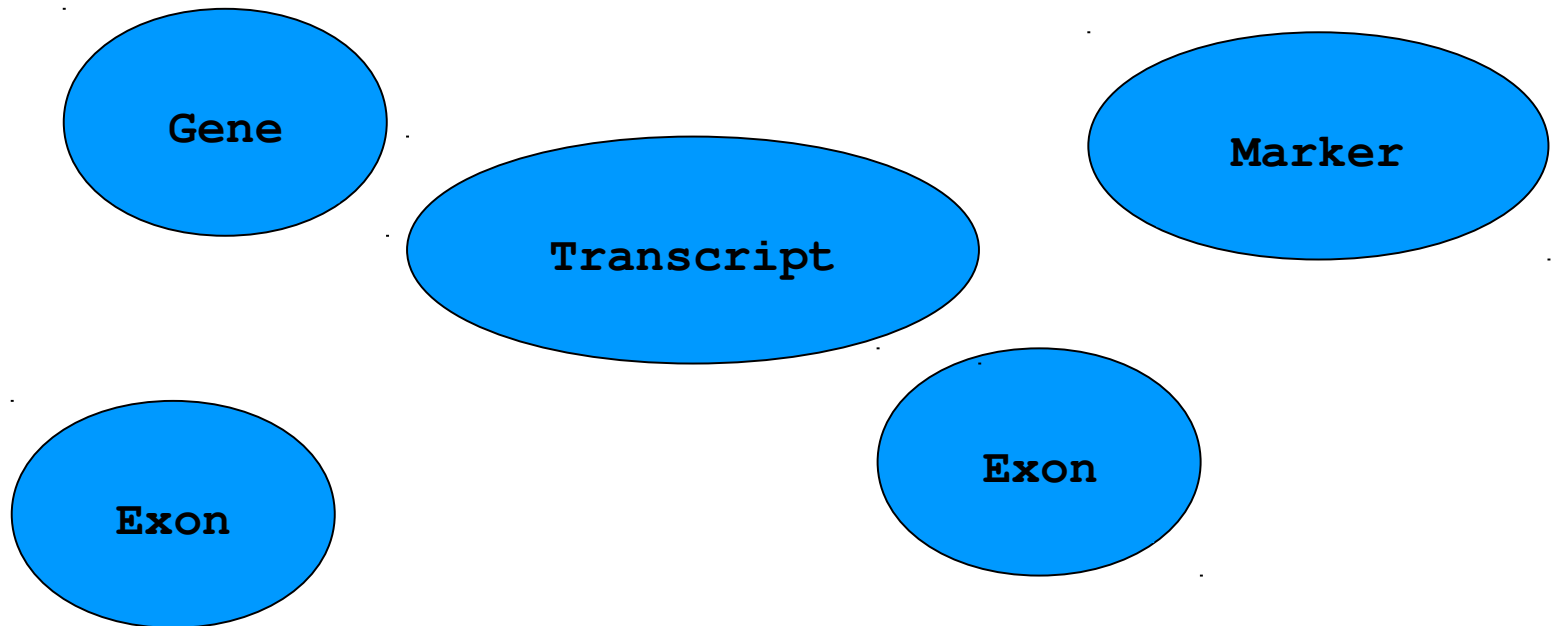
## 2. Object Adaptors

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 **GeneAdaptor** is used to fetch **Gene** objects
- The **ExonAdaptor** 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;
```

Use 'strict' and 'warnings'

```
my $reg = 'Bio::Ensembl::Registry';
```

```
$reg->load_registry_from_db(  
    -host => 'ensembl.ensembl.org',  
    -user => 'anonymous'  
);
```

Change your host and port

```
my $ga = $reg->get_adaptor('human', 'core', 'gene');
```

```
...
```

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



# 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("ENSG00000000123152");
```

# 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:

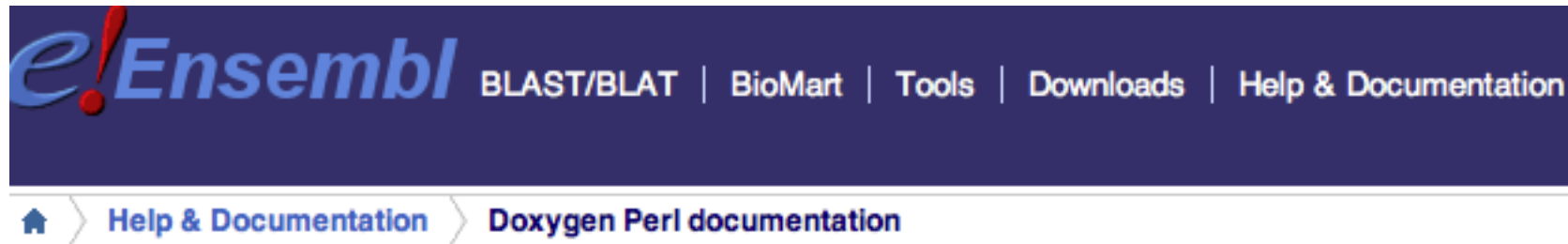
- Object modules start with **Bio::EnsEMBL**
  - Bio::EnsEMBL::Gene for gene objects
  - Bio::EnsEMBL::Exon for exon objects
- ObjectAdaptors start with **Bio::EnsEMBL::DBSQL**
  - Bio::EnsEMBL::DBSQL::GeneAdaptor
  - Bio::EnsEMBL::DBSQL::ExonAdaptor

# 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>



## Doxygen perl module documentation

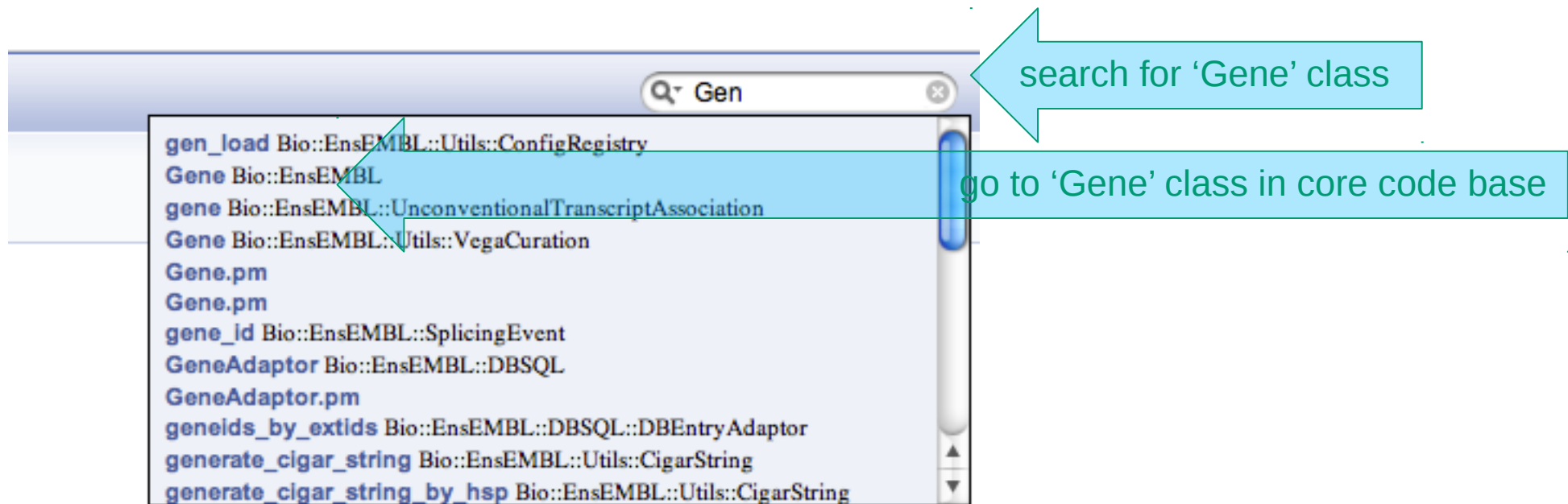
These are the components of the Ensembl code base. Click on a module below for API documentation:

- [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



# Ensembl API Documentation (3)

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

Home > Help & Documentation > API & Software > Doxygen Perl documentation

Overview	Classes	Namespaces	Files	Directory Browser	Related Pages
<div>Class Hierarchy    Class Index    Full Class List    Methods from all packages</div> <div><div>Bio::EnsEMBL::IdMapping::InternalIdMapper::Entry 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 <b>Bio::EnsEMBL::Gene</b> 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</div><div><h2>Bio::EnsEMBL::Gene Class Reference</h2><p>► Inheritance diagram for Bio::EnsEMBL::Gene: List of all members.</p><hr/><h3>Class Summary</h3><h4>Synopsis</h4><pre>my \$gene = Bio::EnsEMBL::Gene-&gt;new(     -START =&gt; 123,     -END   =&gt; 1045,     -STRAND =&gt; 1,     -SLICE =&gt; \$slice );  # print gene information print("gene start:end:strand is "     . join( ":", map { \$gene-&gt;\$_ } qw(start end strand) )     . "\n" );  # set some additional attributes \$gene-&gt;stable_id('ENSG000001'); \$gene-&gt;description('This is the gene description');</pre><h4>Description</h4><p>A representation of a Gene within the Ensembl system. A gene is a set of one or more alternative transcripts.</p></div></div> <div>Bio &gt; EnsEMBL &gt; Gene &gt;</div>					

# 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.

The screenshot shows the Ensembl API documentation for the **Bio::EnsEMBL::Gene** class. The breadcrumb trail at the top reads: Home > Help & Documentation > Doxygen Perl documentation > Ensembl > Bio > EnsEMBL > Gene. Below this is a navigation bar with tabs for Overview, Classes (selected), Namespaces, Files, Directory Browser, and Related Pages. Under the 'Classes' tab, there are sub-tabs: Class Hierarchy, Class Index, Full Class List, and Methods from all packages. The 'Class Hierarchy' sub-tab is active, showing a list of classes on the left. The class **Bio::EnsEMBL::Gene** is highlighted in the list. The main content area is titled **Bio::EnsEMBL::Gene Class Reference** and contains links for an inheritance diagram and a list of all members. Below these links is a 'Class Summary' section with a 'Synopsis' showing a Perl code snippet for creating a Gene object and printing its information. A blue arrow points from the text 'expand to view classes which 'Gene' inherits from' to the 'Inheritance diagram' link.

Help & Documentation > Doxygen Perl documentation > Ensembl > Bio > EnsEMBL > Gene

Overview **Classes** Namespaces Files Directory Browser Related Pages

Class Hierarchy **Class Index** Full Class List Methods from all packages

Bio::EnsEMBL::Utils::Eprof  
Bio::EnsEMBL::Utils::EprofStack  
Bio::EnsEMBL::Utils::Exception  
Bio::EnsEMBL::Exon  
Bio::EnsEMBL::DBSQL::ExonAdaptor  
Bio::EnsEMBL::IdMapping::ExonScoreBuild  
Bio::EnsEMBL::External::ExternalFeatureAdaptor  
Bio::EnsEMBL::DB::ExternalFeatureFactory  
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" );
```



# 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.

Help & Documentation > Doxygen Perl documentation > Ensembl > Bio > EnsEMBL > Gene

Overview	Classes	Namespaces	Files	Directory Browser	Related Pages
Class Hierarchy	Class Index	Full Class List	Methods from all packages		

Bio::EnsEMBL::Utils::Eprof

Bio::EnsEMBL::Utils::EprofStack

Bio::EnsEMBL::Utils::Exception

Bio::EnsEMBL::Exon

Bio::EnsEMBL::DBSQL::ExonAdaptor

Bio::EnsEMBL::IdMapping::ExonScoreBuild

Bio::EnsEMBL::External::ExternalFeatureAdaptor

Bio::EnsEMBL::DB::ExternalFeatureFactory

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](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	External References	Features	
<ul style="list-style-type: none"><li><a href="#">assembly</a></li><li><a href="#">assembly exception</a></li><li><a href="#">coord system</a></li><li><a href="#">data file</a></li><li><a href="#">dna</a></li></ul>	<ul style="list-style-type: none"><li><a href="#">associated group</a></li><li><a href="#">associated xref</a></li><li><a href="#">dependent xref</a></li><li><a href="#">external db</a></li><li><a href="#">external synonym</a></li></ul>	<ul style="list-style-type: none"><li><a href="#">density feature</a></li><li><a href="#">density type</a></li><li><a href="#">ditag</a></li><li><a href="#">ditag feature</a></li><li><a href="#">intron supporting evidence</a></li></ul>	<ul style="list-style-type: none"><li><a href="#">prediction transcript</a></li><li><a href="#">repeat consensus</a></li><li><a href="#">repeat feature</a></li><li><a href="#">simple feature</a></li><li><a href="#">transcript intron supporting evidence</a></li></ul>

# Ensembl Core DB Documentation (2)

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

Fundamental Tables	
• <a href="#">alt allele</a>	• <a href="#">splicing transcript pair</a>
• <a href="#">alt allele attrib</a>	• <a href="#">supporting feature</a>
• <a href="#">alt allele group</a>	• <a href="#">transcript</a>
• <a href="#">analysis</a>	• <a href="#">transcript attrib</a>
• <a href="#">analysis description</a>	• <a href="#">transcript supporting feature</a>
• <a href="#">attrib type</a>	• <a href="#">translation</a>
• <a href="#">dna align feature</a>	• <a href="#">translation attrib</a>
• <a href="#">exon</a>	
• <a href="#">exon transcript</a>	
• <a href="#">gene</a>	
• <a href="#">gene attrib</a>	
• <a href="#">protein align feature</a>	
• <a href="#">protein feature</a>	
• <a href="#">splicing event</a>	
• <a href="#">splicing event feature</a>	

# Ensembl Core DB Documentation (3)

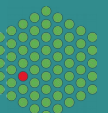
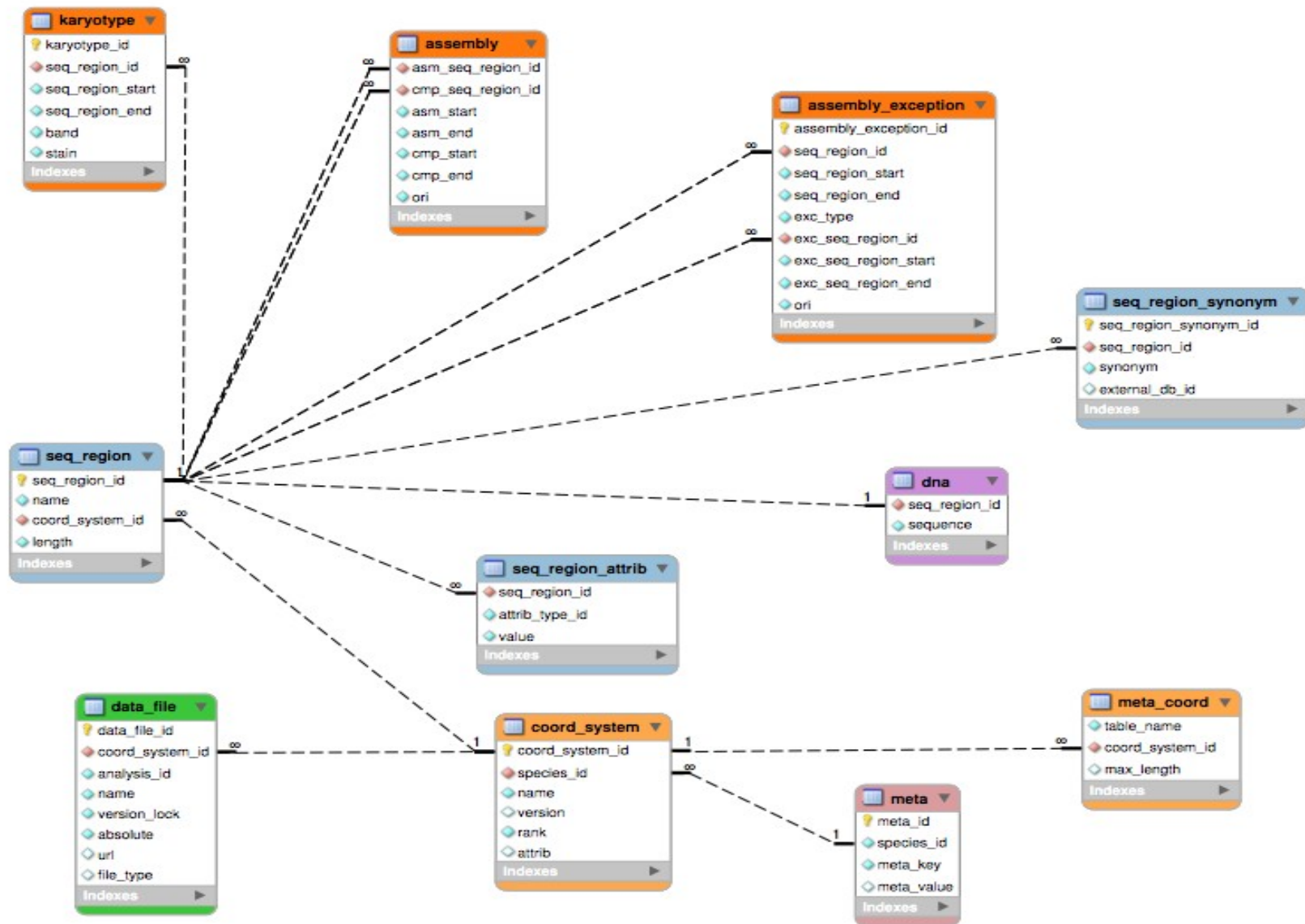
Click on the 'show columns' link to the right to expand a list of gene table columns, their types, descriptions and indices over the columns.

**gene** Hide columns | [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 <a href="#">analysis</a> table.
seq_region_id	INT(10)		Foreign key references to the <a href="#">seq_region</a> 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)



# 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



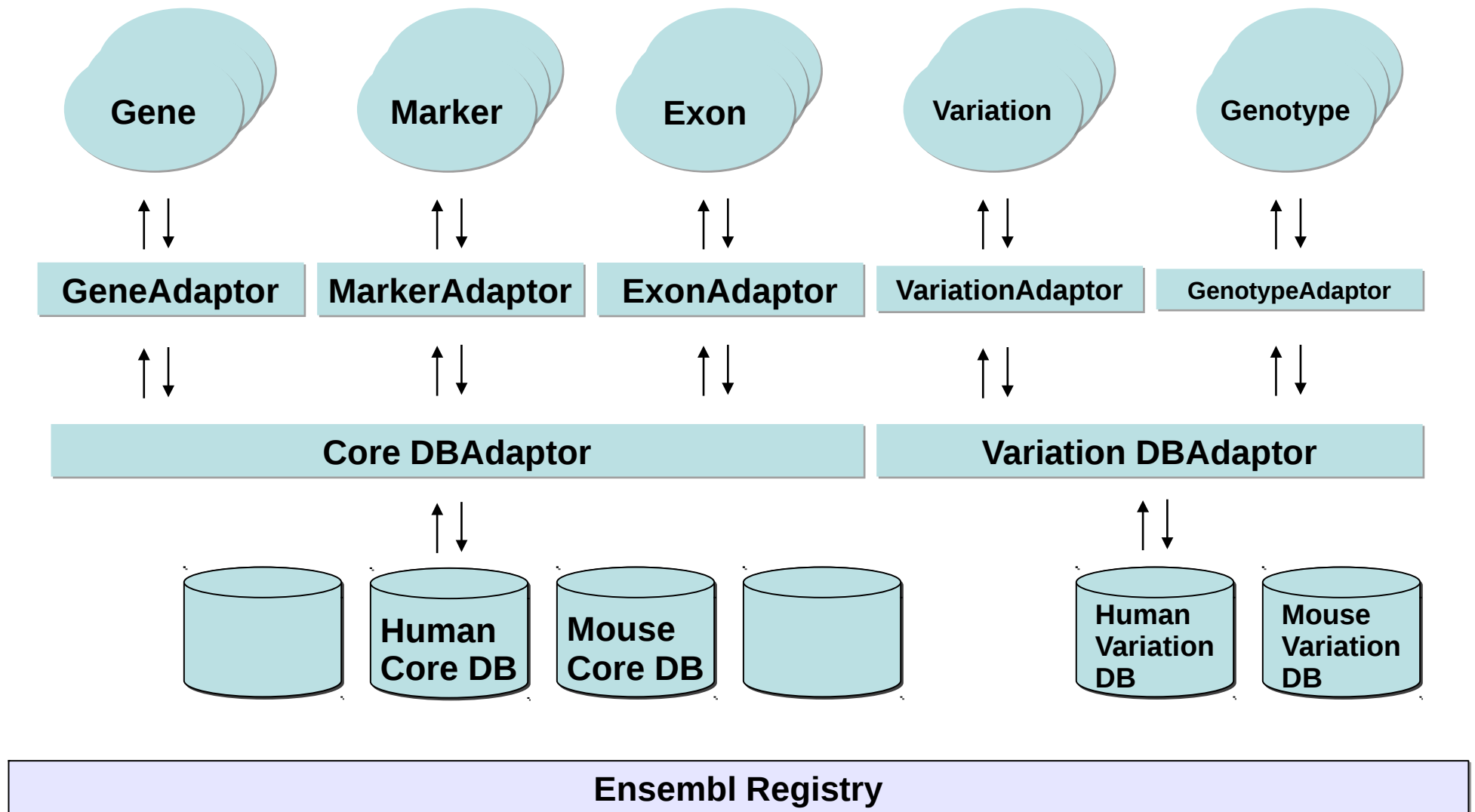
# The Registry

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 => '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.

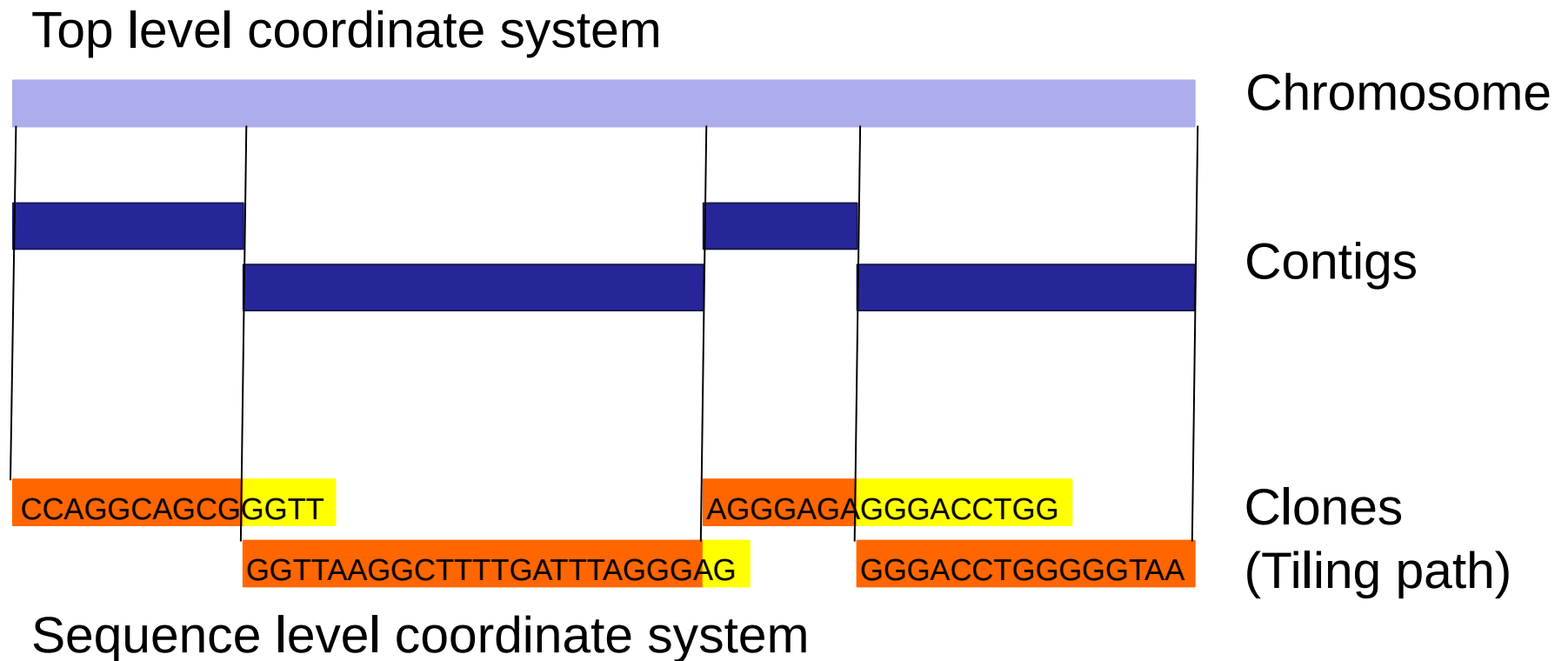
*Hint: Have a look at the Doxygen documentation for the Registry object and method `load_registry_from_db` (<http://www.ensembl.org/info/docs/Doxygen/index.html>).*

# 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

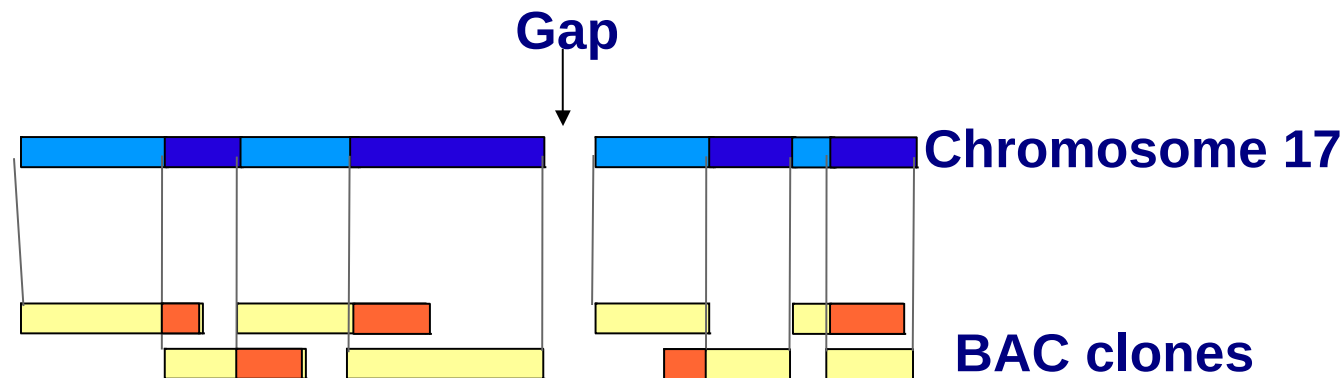
# Coordinate Systems (1)

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



# 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 => 'ensembl.db.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.

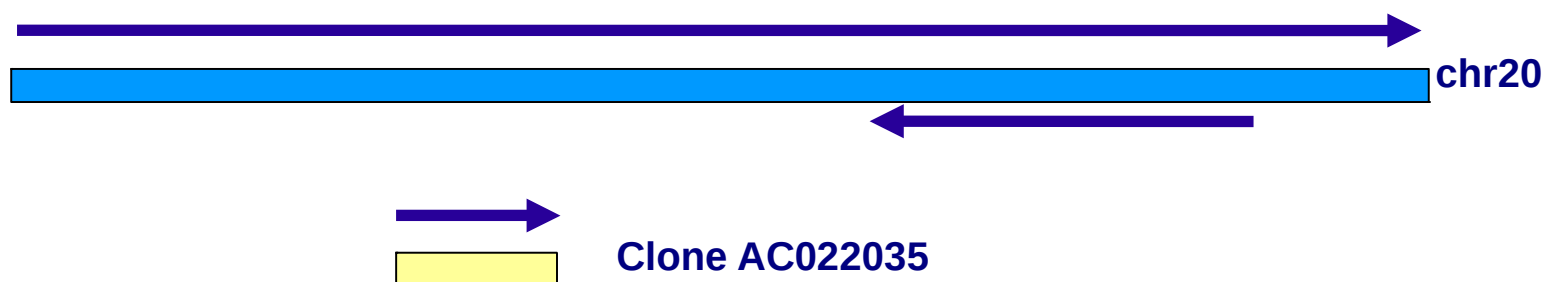
Locus-Reference Genes, used in clinical tests

# 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

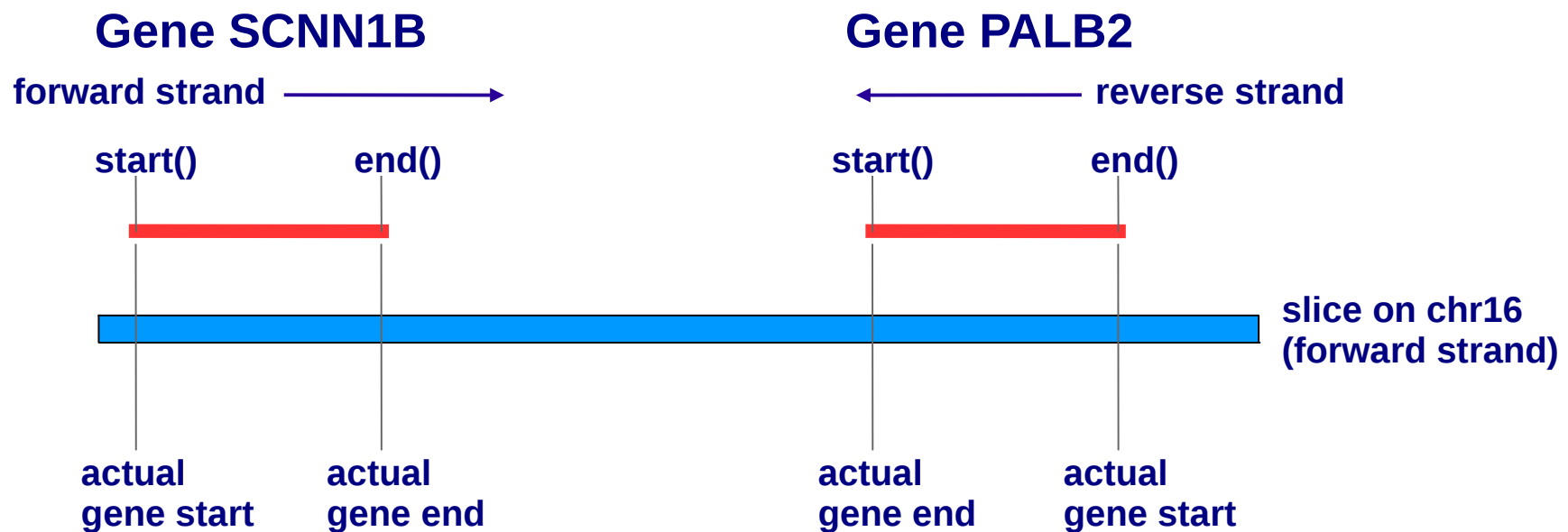
- a. Introduction
- b. Data objects & object adaptors
- c. Ensembl documentation
- d. The Registry & Ensembl API script design
- e. Coordinate systems & slices
- f. **Features**
- g. Genes, transcripts, exons & translations
- h. External references

# 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:  $start < end$

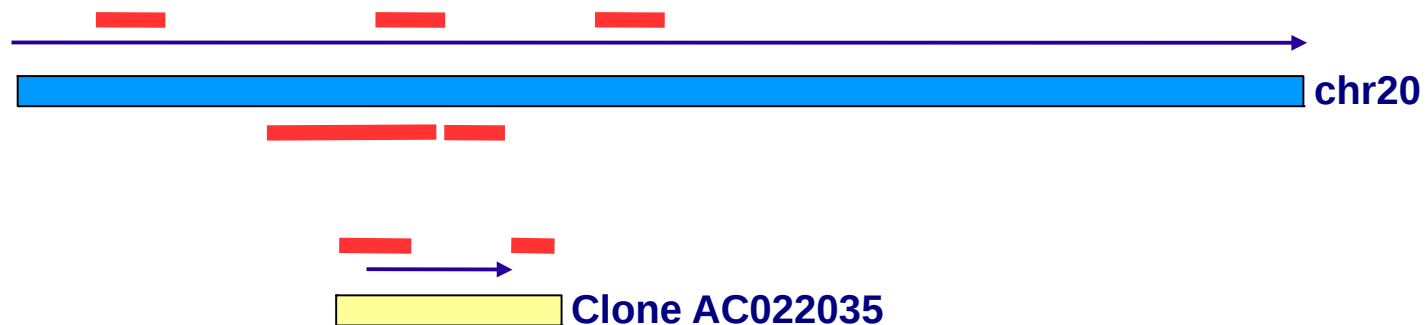


# Features (2)

*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                             |
|--------------------------------------|-----------------------------------------------|
| <b>Gene, Transcript, Exon</b>        | <b>Ensembl gene models</b>                    |
| PredictionTranscript, PredictionExon | Genscan gene models                           |
| DNAAlignFeature, ProteinAlignFeature | cDNAs, proteins                               |
| <b>RepeatFeature</b>                 | <b>repeats</b>                                |
| MarkerFeature                        | markers                                       |
| OligoFeature                         | microarray probes                             |
| KaryotypeBandFeature                 | 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)

|                          |             |     |                                                                 |
|--------------------------|-------------|-----|-----------------------------------------------------------------|
| <b>seq_region_id</b>     | INT(10)     |     | Foreign key references to the <a href="#">seq_region</a> table. |
| <b>seq_region_start</b>  | INT(10)     |     | Sequence start position.                                        |
| <b>seq_region_end</b>    | INT(10)     |     | Sequence end position.                                          |
| <b>seq_region_strand</b> | TINYINT(1)  | '1' | Sequence region strand: 1 - forward; -1 - reverse.              |
| <b>hit_start</b>         | INT(10)     |     | Alignment hit start position.                                   |
| <b>hit_end</b>           | INT(10)     |     | Alignment hit end position.                                     |
| <b>hit_name</b>          | VARCHAR(40) |     | Alignment hit name.                                             |

# Exercise 4

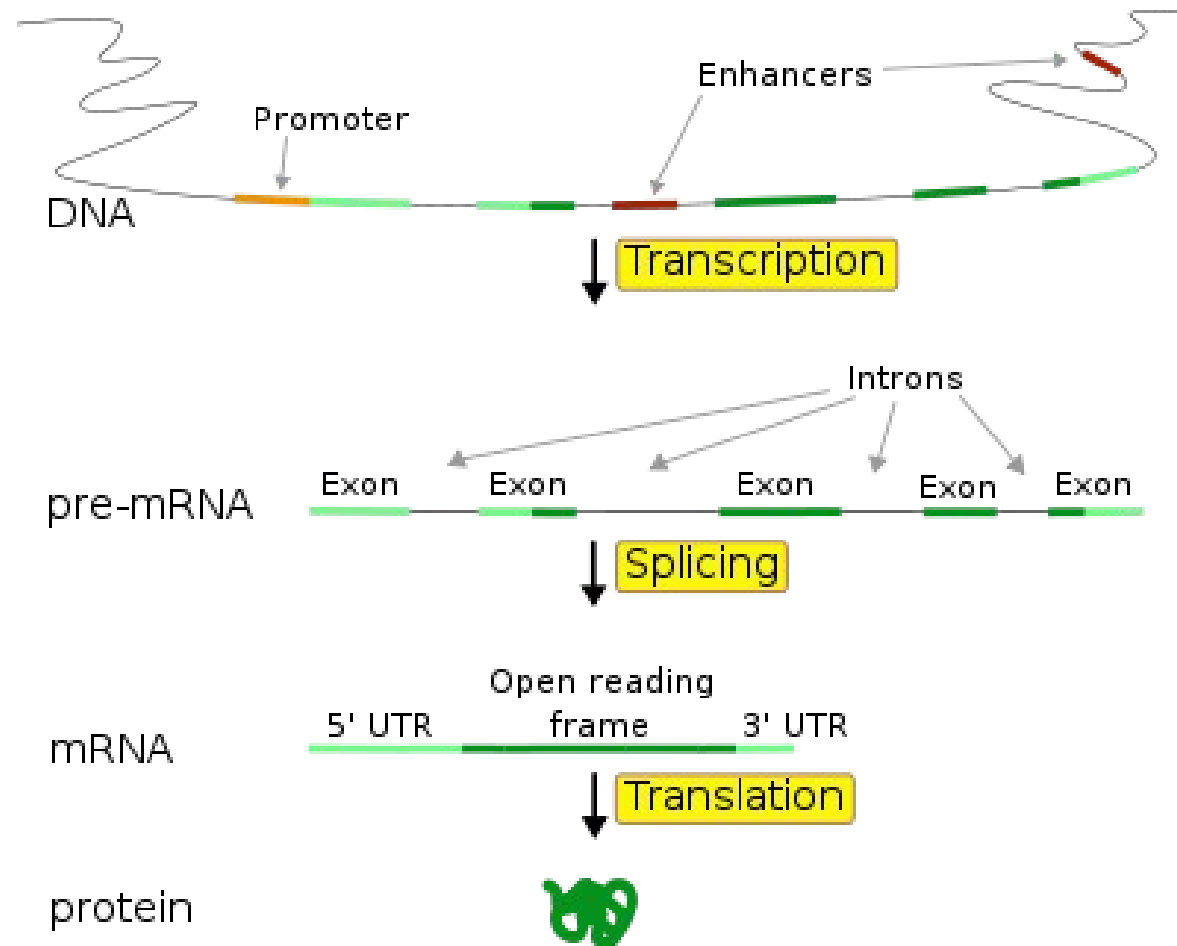
- (a) Get all the repeat features from chromosome 20:1-500kb. Print out the name and position of each on the chromosome and the total number.
  - hint: create a slice, retrieve repeat features on this slice
- (b) Find which genomic region the RefSeq dna entry NM\_000059.3 was mapped to. Print the name 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



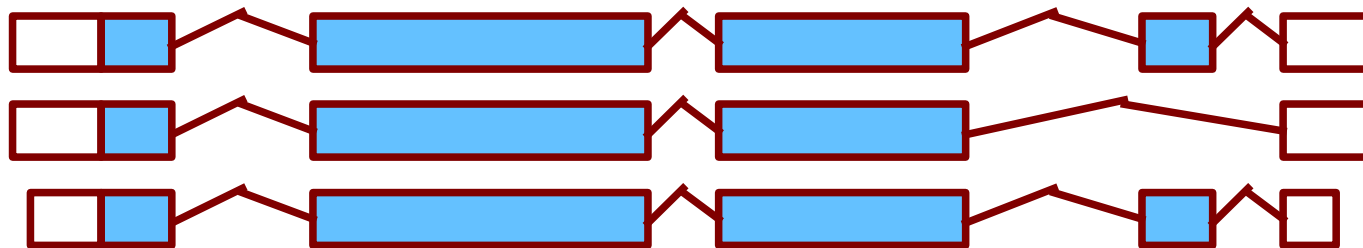
# Genes, Transcripts and Exons

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.
```



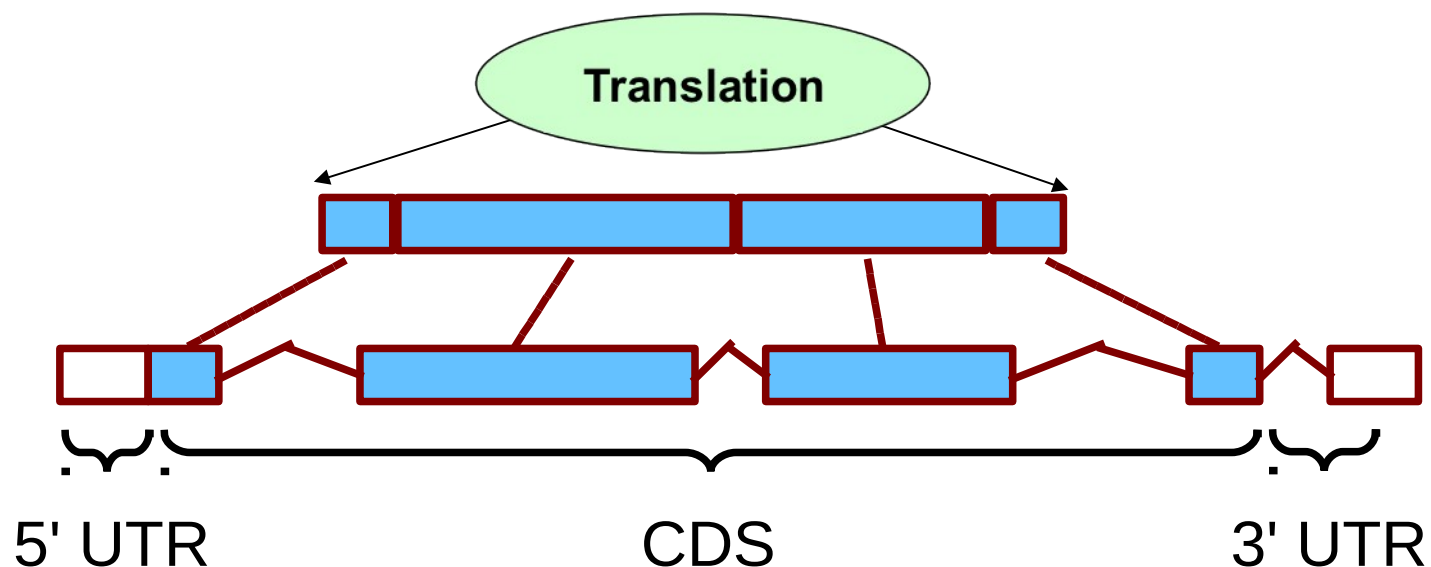
# Translations

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 :

MADPGDPPESESESVFPREVGLFADSYSEKSQFCFCGHVLTITQNFGRSLGVAARVWDAALSLCNYFESQNVDFR  
GKKVIELGAGTGIVGILAAALQGAYGLVRETEDDVIEQELWRGMRGACGHALSMSTMTPWESIKGSSVRGGCYHH

# 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](#)

Show 

All ▾

 entries

Show/hide columns (2 hidden)

Filter

| Name ▾      | Transcript ID ▾                 | Length ▾ | Protein ▾                       | Biotype ▾                  | CCDS ▾                    | Flags ▾                       |
|-------------|---------------------------------|----------|---------------------------------|----------------------------|---------------------------|-------------------------------|
| CSNK2A1-002 | <a href="#">ENST00000217244</a> | 4416 bp  | 391 aa ( <a href="#">view</a> ) | <div></div> Protein coding | <a href="#">CCDS13003</a> | <a href="#">GENCODE basic</a> |
| CSNK2A1-001 | <a href="#">ENST00000349736</a> | 4299 bp  | 391 aa ( <a href="#">view</a> ) | <div></div> Protein coding | <a href="#">CCDS13003</a> | <a href="#">GENCODE basic</a> |
| CSNK2A1-003 | <a href="#">ENST00000400217</a> | 1451 bp  | 255 aa ( <a href="#">view</a> ) | <div></div> Protein coding | <a href="#">CCDS13004</a> | <a href="#">GENCODE basic</a> |
| CSNK2A1-201 | <a href="#">ENST00000619188</a> | 2606 bp  | 397 aa ( <a href="#">view</a> ) | <div></div> Protein coding | -                         | <a href="#">GENCODE basic</a> |
| CSNK2A1-009 | <a href="#">ENST00000400227</a> | 1499 bp  | 385 aa ( <a href="#">view</a> ) | <div></div> Protein coding | -                         | <a href="#">GENCODE basic</a> |

(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?

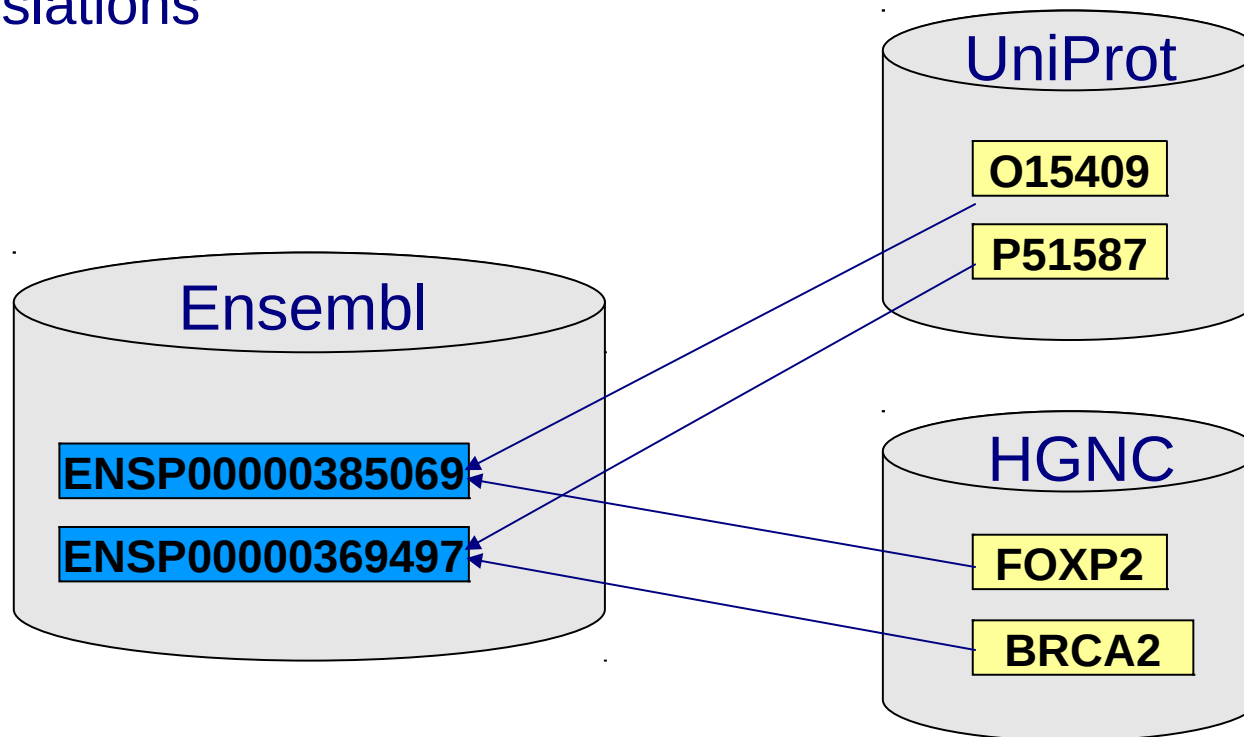
# Outline

- 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";
```

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

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

```
my $all_xrefs = $gene->get_all_DBLinks;
```

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

```
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";  
}
```

# 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');
```

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

```
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";  
}  
}
```

# 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](mailto: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](mailto:helpdesk@ensembl.org)

# Ensembl Acknowledgements

## The Entire Ensembl Team

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

<sup>1</sup>European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Genome Campus, Hinxton, Cambridge CB10 1SD, UK and <sup>2</sup>Wellcome Trust Sanger Institute, Wellcome Genome Campus, Hinxton, Cambridge, CB10 1SA, UK

## Funding



EMBL



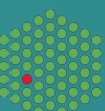
National  
Human Genome  
Research Institute



Open Targets



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

The screenshot shows the Ensembl API documentation page for 'Namespaces'. The page has a navigation bar with links like 'BLAST/BLAT', 'BioMart', 'Tools', 'Downloads', 'Help & Documentation', 'Blog', and 'Mirrors'. Below the navigation bar, there's a breadcrumb trail: 'Help & Documentation > API & Software > Doxygen Perl documentation'. The main content area has tabs for 'Overview', 'Classes', 'Namespaces', 'Files', 'Directory Browser', and 'Related Pages'. The 'Namespaces' tab is selected, showing a 'Namespace List'. On the left, there's a sidebar with a 'Namespace List' section. The main content area lists all the namespaces used in the packages, starting with 'Bio'. The list includes: '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', and 'Bio::EnsEMBL::Utils::IO'. Red arrows point from the text on the right to 'Bio::EnsEMBL' and 'Bio::EnsEMBL::DBSQL' in the list.

| Namespace                                  |
|--------------------------------------------|
| 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

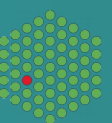
| Attribute | Example value(s)                       | Method(s)                              |
|-----------|----------------------------------------|----------------------------------------|
| name      | chromosome, scaffold, contig,<br>clone | <code>\$coordsystem-&gt;name</code>    |
| version   | GRCh37, NCBI36, NCBIM37                | <code>\$coordsystem-&gt;version</code> |

# Appendix – Feature Methods

| Attribute   | Example value(s)                                      | Method(s)                                                                                                                                                                                                                                         |
|-------------|-------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name        | AluSp, D1S2217                                        | <code>\$feature-&gt;display_id</code><br>} slice relative<br>} chromosome relative                                                                                                                                                                |
| coordinates |                                                       | <code>\$feature-&gt;seq_region_name</code><br><code>\$feature-&gt;start</code><br><code>\$feature-&gt;end</code><br><code>\$feature-&gt;seq_region_start</code><br><code>\$feature-&gt;seq_region_end</code><br><code>\$feature-&gt;strand</code> |
| sequence    |                                                       | <code>\$feature-&gt;seq</code>                                                                                                                                                                                                                    |
| length      | 399                                                   | <code>\$feature-&gt;length</code>                                                                                                                                                                                                                 |
| slice       | returns Slice object with which feature is associated | <code>\$feature-&gt;slice</code>                                                                                                                                                                                                                  |

# Appendix – Gene Methods

| Attribute   | Example value(s)                     | Method(s)                                       |
|-------------|--------------------------------------|-------------------------------------------------|
| stable ID   | ENSG00000139618                      | <code>\$gene-&gt;stable_id</code>               |
| name        | BRCA2                                | <code>\$gene-&gt;external_name</code>           |
| description | breast cancer 2, early onset         | <code>\$gene-&gt;description</code>             |
| biotype     | protein_coding, miRNA                | <code>\$gene-&gt;biotype</code>                 |
| analysis    | ensembl, havana, ensembl_havana_gene | <code>\$gene-&gt;analysis-&gt;logic_name</code> |
| status      | KNOWN, NOVEL                         | <code>\$gene-&gt;status</code>                  |



# Appendix - Transcript Methods

| Attribute                         | Example value(s)                                                           | Method(s)                                             |
|-----------------------------------|----------------------------------------------------------------------------|-------------------------------------------------------|
| stable ID                         | ENST00000380152                                                            | <code>\$transcript-&gt;stable_id</code>               |
| name                              | BRCA2-001                                                                  | <code>\$transcript-&gt;external_name</code>           |
| biotype                           | protein_coding<br>lncRNA<br>antisense_intronic<br>antisense_intronic_decay | <code>\$transcript-&gt;biotype</code>                 |
| analysis                          | ensembl, havana<br>ensembl_havana_transcript                               | <code>\$transcript-&gt;analysis-&gt;logic_name</code> |
| status                            | KNOWN, NOVEL                                                               | <code>\$transcript-&gt;status</code>                  |
| CDS (spliced<br>sequence, no UTR) | ATGCCTATTGGATCCAAAGAGAGGC...                                               | <code>\$transcript-&gt;translateable_seq</code>       |



# Appendix - Translation Methods

| Attribute | Example value(s)             | Method(s)                                |
|-----------|------------------------------|------------------------------------------|
| stable id | ENSP00000369497              | <code>\$translation-&gt;stable_id</code> |
| length    | 3418                         | <code>\$translation-&gt;length</code>    |
| sequence  | MPIGSKERPTFFEIFKTRCNKADLG... | <code>\$translation-&gt;seq</code>       |