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

- 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.

```perl
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

Ensembl REST API Endpoints

**Comparative Genomics**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET genetree/id/id</td>
<td>Retrieves Gene Tree dumps for a given Gene Tree stable identifier</td>
</tr>
<tr>
<td>GET homology/id/id</td>
<td>Retrieves homology information by ensemble gene id</td>
</tr>
<tr>
<td>GET homology/symbol/species/symbol</td>
<td>Retrieves homology information by symbol</td>
</tr>
</tbody>
</table>

**Cross References**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET xrefs/id/id</td>
<td>Perform lookups of Ensembl Identifiers and retrieve their external cross references in other databases</td>
</tr>
<tr>
<td>GET xrefs/name/species/name</td>
<td>Performs a lookup based upon the primary accession or display label of an external reference and returning the information we hold about the entry</td>
</tr>
<tr>
<td>GET xrefs/symbol/species/symbol</td>
<td>Looks up an external symbol and returns all Ensembl objects linked to it. This can be a display name for a gene/transcript/translation, a synonym or an externally linked reference. If a Gene's transcript is linked to the supplied symbol the service will return both Gene and Transcript (if supports transient link).</td>
</tr>
</tbody>
</table>

**Information**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET assembly/info/species</td>
<td>Returns information about the current available assemblies in this given species</td>
</tr>
<tr>
<td>GET assembly/info/species/region_name</td>
<td>Returns information about the given toplevel sequence region given to this endpoint</td>
</tr>
</tbody>
</table>

http://beta.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 1.2.3
wget http://bioperl.org/DIST/old_releases/bioperl-1.2.3.tar.gz

# untar both
tar zxvf ensembl-api.tar.gz
tar zxvf bioperl-1.2.3.tar.gz

# open up .bashrc or .profile and add the following
PERL5LIB=${PERL5LIB}:${HOME}/src/bioperl-1.2.3
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

- Ensembl tarball from http://cvs.sanger.ac.uk
  - Watch out for & character in Linux shells
- CVS checkout from cvs.sanger.ac.uk
- Ensembl Virtual Machine
  - 64bit only in OVA format
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.

1. **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 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:

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

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

GET (no arg.):

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

SET (new value):

```perl
$gene->stable_id(“ENSG00000000123152”);
```
# 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
  - the GeneAdaptor is a Bio::EnsEMBL::DBSQL::GeneAdaptor
  - The ExonAdaptor is a Bio::EnsEMBL::DBSQL::ExonAdaptor

Naming conventions for the methods:

- For retrieving objects from the ObjectAdaptors:
  - fetch_by_ and fetch_all_by_
- For retrieving Objects attributes:
  - get_Object and get_all_Objects
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

Core code base
Search for classes or methods

search for ‘Gene’ class

go to ‘Gene’ class in core code
Ensembl API Documentation (3)

Breadcrumbs at the top of the page help you keep track of which code base you’re in.
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.
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? What variable type does this method return?
Clicking on a method takes you to a description with associated arguments, return types and exceptions.

```java
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 analogous 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.
Clicking on **Code:** shows the method’s implementation

```perl
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

Introduction

This document gives a high-level description of the tables that make up the EnaEMBL core schema. Tables are grouped into logical groups, and 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 important columns in some of the tables are discussed, this document makes no attempt to enumerate all of the names, types and contents of the table descriptions are given at the end of this document; these are linked to from the table description where appropriate.

Different tables are populated throughout the gene build process:

<table>
<thead>
<tr>
<th>Step</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Create empty schema, populate meta table</td>
</tr>
<tr>
<td>1</td>
<td>Load DNA - populates dna, clone, contig, chromosome, assembly tables</td>
</tr>
<tr>
<td>2</td>
<td>Analyze DNA (raw computes) - populates genomic feature/analysis tables</td>
</tr>
<tr>
<td>3</td>
<td>Build genes - populates exon, transcript, etc. gene-related tables</td>
</tr>
<tr>
<td>4a</td>
<td>Analyze genes - populate protein_feature, xref tables, interpro</td>
</tr>
<tr>
<td>4b</td>
<td>ID mapping</td>
</tr>
</tbody>
</table>

This document refers to version 65 of the EnaEMBL core schema.

List of the tables:

Fundamental Tables
Ensembl Core DB Documentation (2)

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

**Fundamental Tables**

- assembly
- assembly_exception
- attrib_type
- coord_system
- dna
- dnac
- exon
- exon_transcript
- gene
- gene_atrib
- karyotype
- meta
- meta_coord
- operon
- operon_transcript
- operon_transcript_gene
- seq_region
- seq_region_atrib
- transcript
- transcript_atrib
- translation
- translation_atrib
- unconventional_transcript_association
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.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene_id</td>
<td>INT(10)</td>
<td></td>
<td>Primary key, internal identifier.</td>
</tr>
<tr>
<td>biotype</td>
<td>VARCHAR(40)</td>
<td></td>
<td>Biotype, e.g. protein_coding.</td>
</tr>
<tr>
<td>analysis_id</td>
<td>SMALLINT</td>
<td></td>
<td>Foreign key references to the analysis table.</td>
</tr>
<tr>
<td>seq_region_id</td>
<td>INT(10)</td>
<td></td>
<td>Foreign key references to the seq_region table.</td>
</tr>
<tr>
<td>seq_region_start</td>
<td>INT(10)</td>
<td></td>
<td>Sequence start position.</td>
</tr>
<tr>
<td>seq_region_end</td>
<td>INT(10)</td>
<td></td>
<td>Sequence end position.</td>
</tr>
<tr>
<td>seq_region_strand</td>
<td>TINYINT(2)</td>
<td></td>
<td>Sequence region strand: 1 - forward; -1 - reverse.</td>
</tr>
<tr>
<td>display_xref_id</td>
<td>INT(10)</td>
<td></td>
<td>External reference for EnsEMBL web site. Foreign key references to the xref table.</td>
</tr>
<tr>
<td>source</td>
<td>VARCHAR(20)</td>
<td></td>
<td>e.g. ensembl, havana etc.</td>
</tr>
<tr>
<td>status</td>
<td>ENUM('KNOWN', 'NOVEL', 'PUTATIVE', 'PREDICTED')</td>
<td></td>
<td>Status, e.g. KNOWN, NOVEL, PUTATIVE, PREDICTED.</td>
</tr>
</tbody>
</table>
Ensembl Core DB Documentation (4)

- expand or collapse column information for all tables

**Fundamental Tables**

A PDF document of the schema is available [here](#).

- DB schema diagram is available for each group of tables

**assembly**

The assembly table states, which parts of seq_regions are exactly equal. It enables to transform coordinates between chromosomes out of supercontigs. It allows you to artificially chunk chromosome sequence into smaller parts. UCSC or NCBI (depending which assembly you are using). Each row represents a component, e.g. a contig, (c that is in the path is delimited by fields cmp_start and cmp_end (start < end), and the absolute position within the chromosome.

See also:

- seq_region
- supercontigs
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?