Take a REST from manual searching
Ensembl, programmatically

Matthew Laird
matthew.laird@ebi.ac.uk
helpdesk@ensembl.org
Questions?

- We’ve muted all the mics
- Ask questions in the Chat box in the webinar interface
- My colleague, Emily, will answer your questions in the Chat box
- There’s no threading so please respond with @name
Contents

• What is Ensembl?
• Methods to access Ensembl data?
• The Ensembl REST server documentation.
• How to query the REST server.
• An example.
• Where to go for help and documentation.
Ensembl Features

- Gene builds for ~70 species
- Gene trees
- Regulatory build (ENCODE)
- Variation display and VEP
- Display of user data
- BioMart (data export)
- Programmatic access via the perl APIs
- REST API
- Completely Open Source
Release cycle

- 2-3 months
- New/updated interfaces
- Updated regulation data
- Updated variation data
- Comparison on new genes and genomes
- Updated gene sets
- New genome assemblies
- Underlying software updates
- Updated variation data
- Updated gene sets
- New genome assemblies
Ensembl vs Ensembl Genomes

Ensembl and Ensembl Genomes contain different sets of species

**Ensembl**
- Vertebrate
- C. elegans
- D. melanogaster
- S. cerevisiae

**Ensembl Genomes**
- Bacteria
- Fungi
- Metazoa
- Plants
- Protists
Ensembl Data Model

- Primary feature types of **Genes**, **Transcript**, and **Exons**
- A **Gene** is a set of alternatively spliced **Transcripts**
- A **Transcript** is a set of **Exons**
- **Peptides** are not stored in the database, they are computed on the fly using **Transcript** objects
Ensembl Data Model

- Features have a defined location on the genome
- Start and end are always plotted on the forward strand
- start < end
Ensembl access methods

- One by one
- Main browser
  - Mobile site
- BioMart
- REST API
- VEP
- Perl API
- MySQL
- Groups
- FTP
- Whole genome
Ensembl REST API

- Language agnostic access to Ensembl datasets
- Only a fraction of the functionality of the Perl API is exposed

http://rest.ensembl.org
Available REST services

http://rest.ensembl.org - Vertebrate + C. elegans, D. melanogaster, S. cerevisiae

http://grch37.rest.ensembl.org - Human GRCh37 assembly

http://e87.rest.ensembl.org - Archives, e87 is first available

http://rest.ensemblgenomes.org - All other species
What is a REST API?

REpresentational State Transfer. It describes how one system can communicate state with another.

Typically over HTTP(S), providing a machine readable, language agnostic method to access remote data or services.

http://rest.ensembl.org/datalwant

Gene: IRAK4, start: 43758944, end: 43789543, ...
Our first REST call

http://rest.ensembl.org/info/ping?content-type=application/json

{
  ping: 1
}


A REST request

rest.ensembl.org/lookup/id/ENSG00000157764?content-type=application/json

```json
{
    "source": "ensembl_havana",
    "object_type": "Gene",
    "logic_name": "ensembl_havana_gene",
    "version": 12,
    "species": "homo_sapiens",
    "description": "B-Raf proto-oncogene, serine/threonine kinase [Source:HGNC Symbol;Acc:HGNC:1097]",
    "display_name": "BRAF",
    "assembly_name": "GRCh38",
    "biotype": "protein_coding",
    "end": 140924764,
    "seq_region_name": "7",
    "db_type": "core",
    "strand": -1,
    "id": "ENSG00000157764",
    "start": 140719327
}
```
Endpoint Documentation

Ensembl REST API Endpoints

### Archive

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET archive/id/id</td>
<td>Uses the given identifier to return the archived sequence</td>
</tr>
<tr>
<td>POST archive/id</td>
<td>Retrieve the archived sequence for a set of identifiers</td>
</tr>
</tbody>
</table>

### Comparative Genomics

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET genetree/id/id</td>
<td>Retrieves a gene tree for a gene tree stable identifier</td>
</tr>
<tr>
<td>GET genetree/member/id/id</td>
<td>Retrieves the gene tree that contains the gene / transcript / translation stable identifier</td>
</tr>
<tr>
<td>GET genetree/member/symbol/species/symbol</td>
<td>Retrieves the gene tree that contains the gene identified by a symbol</td>
</tr>
<tr>
<td>GET alignment/region/species/region</td>
<td>Retrieves genomic alignments as separate blocks based on a region and species</td>
</tr>
<tr>
<td>GET homology/id/id</td>
<td>Retrieves homology information (orthologs) by Ensembl gene id</td>
</tr>
<tr>
<td>GET homology/symbol/species/symbol</td>
<td>Retrieves homology information (orthologs) by symbol</td>
</tr>
</tbody>
</table>

### Cross References

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 (it supports transient links).</td>
</tr>
<tr>
<td>GET xrefs/id/id</td>
<td>Perform lookups of Ensembl Identifiers and retrieve their external 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</td>
</tr>
</tbody>
</table>

http://rest.ensembl.org
Functional groupings

- Comparative Genomics
- Cross References
- Information
- Lookup
- Mapping
- Ontology & Taxonomy
- Sequence
- Variation
- VEP, etc...

Ensembl REST API Endpoints

**Archive**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET archive/id/id</td>
<td>Uses the given identifier to return the archived sequence</td>
</tr>
<tr>
<td>POST archive/id</td>
<td>Retrieve the archived sequence for a set of identifiers</td>
</tr>
</tbody>
</table>

**Comparative Genomics**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET genetree/id/id</td>
<td>Retrieves a gene tree for a gene tree stable identifier</td>
</tr>
<tr>
<td>GET genetree/member/id/id</td>
<td>Retrieves the gene tree that contains the gene / transcript / translation stable identifier</td>
</tr>
<tr>
<td>GET genetree/member/symbol/species/symbol</td>
<td>Retrieves the gene tree that contains the gene identified by a symbol</td>
</tr>
<tr>
<td>GET alignment/region/species/region</td>
<td>Retrieves genomic alignments as separate blocks based on a region and species</td>
</tr>
<tr>
<td>GET homology/id/id</td>
<td>Retrieves homology information (orthologs) by Ensembl gene id</td>
</tr>
<tr>
<td>GET homology/symbol/species/symbol</td>
<td>Retrieves homology information (orthologs) by symbol</td>
</tr>
</tbody>
</table>

**Cross References**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET refs/symbol/species/symbol</td>
<td>Looks up an external symbol and returns all Ensembl objects linked to it.</td>
</tr>
<tr>
<td>GET refs/id/id</td>
<td>Perform lookups of Ensembl Identifiers and retrieve their external references in other databases</td>
</tr>
<tr>
<td>GET refs/name/species/name</td>
<td>Performs a lookup based upon the primary accession or display label of an external reference and returning the information</td>
</tr>
</tbody>
</table>
## Endpoint Documentation

### GET lookup/id:id

Find the species and database for a single identifier e.g. gene, transcript, protein

#### Parameters

##### Required

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>An Ensembl stable ID</td>
<td>-</td>
<td>ENSG00000157784</td>
</tr>
</tbody>
</table>

##### Optional

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>String</td>
<td>Name of the callback subroutine to be returned by the requested JSONP response. Required ONLY when using JSONP as the serialisation method. Please see the user guide.</td>
<td>-</td>
<td>randomlygeneratedname</td>
</tr>
<tr>
<td>db_type</td>
<td>String</td>
<td>Restrict the search to a database other than the default. Useful if you need to use a DB other than core</td>
<td>-</td>
<td>core otherfeatures</td>
</tr>
<tr>
<td>expand</td>
<td>Boolean(0,1)</td>
<td>Expands the search to include any connected features. e.g. If the object is a gene, its transcripts, translations and exons will be returned as well.</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
**GET lookup/id:id**

Find the species and database for a single identifier e.g. gene, transcript, protein

### Parameters

#### Required

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>An Ensembl stable ID</td>
<td>-</td>
<td>ENSG00000157764</td>
</tr>
</tbody>
</table>

#### Optional

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>String</td>
<td>Name of the callback subroutine to be returned by the requested JSONP response. Required ONLY when using JSONP as the serialisation method. Please see the user guide.</td>
<td>-</td>
<td>randomlygeneratedname</td>
</tr>
<tr>
<td>db_type</td>
<td>String</td>
<td>Restrict the search to a database other than the default. Useful if you need to use a DB other than core</td>
<td>-</td>
<td>core otherfeatures</td>
</tr>
<tr>
<td>expand</td>
<td>Boolean(0,1)</td>
<td>Expands the search to include any connected features. e.g. If the object is a gene, its transcripts, translations and exons will be returned as well.</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
**Endpoint Documentation**

**GET lookup/id/:id**

Find the species and database for a single identifier e.g. gene, transcript, protein

### Parameters

#### Required

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>An Ensembl stable ID</td>
<td>-</td>
<td>ENSG00000157764</td>
</tr>
</tbody>
</table>

#### Optional

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>String</td>
<td>Name of the callback subroutine to be returned by the requested JSONP response. Required ONLY when using JSONP as the serialisation method. Please see the user guide.</td>
<td>-</td>
<td>randomlygeneratedname</td>
</tr>
<tr>
<td>db_type</td>
<td>String</td>
<td>Restrict the search to a database other than the default. Useful if you need to use a DB other than core</td>
<td>-</td>
<td>core otherfeatures</td>
</tr>
<tr>
<td>expand</td>
<td>Boolean(0,1)</td>
<td>Expands the search to include any connected features. e.g. If the object is a gene, its transcripts, translations and exons will be returned as well.</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

Resource Information

<table>
<thead>
<tr>
<th>Methods</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response formats</td>
<td>json xmljsonp</td>
</tr>
</tbody>
</table>
# GET lookup/id/:id

Find the species and database for a single identifier e.g. gene, transcript, protein

## Parameters

### Required

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>An Ensembl stable ID</td>
<td>-</td>
<td>ENSG00000157764</td>
</tr>
</tbody>
</table>

### Optional

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>String</td>
<td>Name of the callback subroutine to be returned by the requested JSONP response. Required ONLY when using JSONP as the serialisation method. Please see the user guide.</td>
<td>-</td>
<td>randomlygeneratedname</td>
</tr>
<tr>
<td>db_type</td>
<td>String</td>
<td>Restrict the search to a database other than the default. Useful if you need to use a DB other than core.</td>
<td>-</td>
<td>core otherfeatures</td>
</tr>
<tr>
<td>expand</td>
<td>Boolean</td>
<td>Expands the search to include any connected features. e.g. If the object is a gene, its transcripts, translations and exons will be returned as well.</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
### Endpoint Documentation

**GET lookup/id:id**

Find the species and database for a single identifier e.g. gene, transcript, protein

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>An Ensembl stable ID</td>
<td>-</td>
<td>ENSG00000157764</td>
</tr>
</tbody>
</table>

#### Optional

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>db_type</td>
<td>String</td>
<td>Restrict the search to a database other than the default. Useful if you need to use a DB other than core.</td>
<td>-</td>
<td>core otherfeatures</td>
</tr>
<tr>
<td>expand</td>
<td>Boolean[0,1]</td>
<td>Expands the search to include any connected features. e.g. If the object is a gene, its transcripts, translations and exons will be returned as well.</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

Request Example:

```
/llookup/id/ENSG00000157764?expand=1
```
### Endpoint Documentation

Content-type and Accept headers are how servers and clients negotiate what format to send results in.

---

#### GET lookup/id:id

Find the species and database for a single identifier e.g. gene, transcript, protein

**Parameters**

**Required**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>An Ensembl stable ID</td>
<td>-</td>
<td>ENSG00000157764</td>
</tr>
</tbody>
</table>

**Optional**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>String</td>
<td>Name of the callback subroutine to be returned by the requested JSONP response. Required ONLY when using JSONP as the serialisation method. Please see the user guide.</td>
<td>-</td>
<td>randomlygeneratedname</td>
</tr>
<tr>
<td>db_type</td>
<td>String</td>
<td>Restrict the search to a database other than the default. Useful if you need to use a DB other than core</td>
<td>-</td>
<td>core otherfeatures</td>
</tr>
<tr>
<td>expand</td>
<td>Boolean(0,1)</td>
<td>Expands the search to include any connected features. e.g. if the object is a gene, its transcripts, translations and exons will be returned as well.</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

---

**Resource Information**

<table>
<thead>
<tr>
<th>Methods</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response formats</td>
<td>json xml jsonp</td>
</tr>
</tbody>
</table>
Content-type and Accept headers are how servers and clients negotiate what format to send results in.

```
/lookup/id/ENSG00000157764?content-type=application/json
```
use strict;
use warnings;

use HTTP::Tiny;

my $http = HTTP::Tiny->new();

my $server = 'http://rest.ensembl.org';
my $ext = '/lookup/id/ENSG000000157764?expand=1';
my $response = $http->get($server.$ext, {
  headers => { 'Content-type' => 'application/json' }
});

die "Failed!\n" unless $response->{success};

use JSON;
use Data::Dumper;

if(length $response->{content}) {

What does an HTTP response look like?

HTTP/1.1 200 OK
Date: Mon, 24 Apr 2017 01:11:12 GMT
Server: Apache/2.4.11 (Linux)
Last-Modified: Mon, 24 Apr 2017
ETag: "0-23-43bc86a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-type: text/html

<h1>My Home page</h1>
HTTP Status Codes

The server uses HTTP status codes to signal request outcome

http://rest.ensembl.org/thisdoesntexist
## HTTP Status Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>Request was a success</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>Occurs during exceptional circumstances such as the service is unable to find an ID. Check if the response Content-type or Accept was JSON. If so the JSON object is an exception hash with the message keyed under error</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>You are submitting far too many requests and have been temporarily forbidden access to the service. Wait and retry with a maximum of 15 requests per second.</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>Indicates a badly formatted request. Check your URL</td>
</tr>
</tbody>
</table>
## HTTP Status Codes (cont.)

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>408</td>
<td>Timeout</td>
<td>The request was not processed in time. Wait and retry later</td>
</tr>
<tr>
<td>429</td>
<td>Too Many Requests</td>
<td>You have been rate-limited; wait and retry. The headers X-RateLimit-Reset, X-RateLimit-Limit and X-RateLimit-Remaining will inform you of how long you have until your limit is reset and what that limit was. If you get this response and have not exceeded your limit then check if you have made too many requests per second.</td>
</tr>
<tr>
<td>503</td>
<td>Service Unavailable</td>
<td>The service is temporarily down; retry after a pause</td>
</tr>
</tbody>
</table>
# Content Types

The most common content types you’ll use.

<table>
<thead>
<tr>
<th>Resource Information</th>
<th>FASTA</th>
<th>XML</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>text/x-fasta</td>
<td>text/xml</td>
<td>application/json</td>
</tr>
<tr>
<td>Response formats</td>
<td>FASTA</td>
<td>XML</td>
<td>JSON</td>
</tr>
<tr>
<td>fasta</td>
<td>FASTA</td>
<td>XML</td>
<td>JSON</td>
</tr>
<tr>
<td>json</td>
<td>FASTA</td>
<td>XML</td>
<td>JSON</td>
</tr>
<tr>
<td>seqxml</td>
<td>FASTA</td>
<td>XML</td>
<td>JSON</td>
</tr>
<tr>
<td>text</td>
<td>FASTA</td>
<td>XML</td>
<td>JSON</td>
</tr>
<tr>
<td>yamll</td>
<td>FASTA</td>
<td>XML</td>
<td>JSON</td>
</tr>
<tr>
<td>jsonp</td>
<td>FASTA</td>
<td>XML</td>
<td>JSON</td>
</tr>
</tbody>
</table>

- **FASTA**: Sequence serialization format. Only supported on the /sequence endpoint.
- **XML**: A structured text document, allowing the creation of customized markup languages.
- **JSON**: A minimal, readable format for structuring data. Similar to XML but typically more easily readable by humans.
Chaining requests

```
rest.ensembl.org/lookup/symbol/human/ENSG00000157764?content-type=application/json
```

```json
{
  "source": "ensembl_havana",
  "object_type": "Gene",
  "logic_name": "ensembl_havana_gene",
  "version": 13,
  "species": "homo_sapiens",
  "description": "interleukin 1 receptor associated kinase 4 [Source:HGNC Symbol;Acc:HGNC:17967]",
  "display_name": "IRAK4",
  "assembly_name": "GRCh38",
  "biotype": "protein_coding",
  "end": 43789543,
  "seq_region_name": "12",
  "db_type": "core",
  "strand": 1,
  "id": "ENSG00000198001",
  "start": 43758944
}
```
Decoding the response

- In many cases you’ll be using JSON formatted responses
- Most languages have JSON parsers that return the data as a structure

```python
decoded = r.json()

print decoded['id']

> ENSG00000157764
```
Using results in another query

```python
ext = "/sequence/id/{}?expand=1".format(decoded['id'])

r = requests.get(server+ext, headers={"Content-Type": "text/x-fasta"})

if not r.ok:
    r.raise_for_status()

print r.text[0:200]

>ENSG00000198001 chromosome:GRCh38:12:43758944:43789543:1
CCGCCCCCTTGCCGCCGCTTCTCTAGTTCGGCTTGTTCTCTGTCGCCGGGCTTACGCC
CGCGCCGGGGAGGTAAGGCATTCTGCTCCCCTTGATCGCTGGTCAGCAAGG
CGCTTCCAGGCCTCGCAGAC
```
Live example

- Fetch a gene based on **symbol**
- Extract Ensembl **Stable ID** from the JSON record
- Use the **Stable ID** to fetch the sequence in **FASTA** format
- Print the first **201 characters** of the sequence
HTTP Methods - GET vs POST

GET http://rest.ensembl.org/lookup/ENSG00000157764

POST http://rest.ensembl.org/lookup/

```json
{ "ids" : ["ENSG00000157764", "ENSG00000248378"] }
```
Useful Tools – Postman
Using POST

```python
import requests, sys

server = "http://rest.ensembl.org"
ext = "/lookup/id"
headers={ "Content-Type" : "application/json", "Accept" : "application/json"}
r = requests.post(server+ext, headers=headers, data='{"ids" : ["ENSG00000157764", "ENSG00000248378" ] }')

# error checking remove for space

decoded = r.json()
print repr(decoded)
```
Using POST

Notice, this endpoint returns a dictionary of dictionaries

```json
{
    "ENSG00000157764": {
        "source": "ensembl_havana",
        "object_type": "Gene",
        ...
    },
    "ENSG00000248378": {
        "source": "havana",
        "object_type": "Gene",
        ...
    }
}
```
Rate limiting

Requests are rate limited to prevent a single user from monopolising the resources.

X-RateLimit-Limit: 55000
X-RateLimit-Reset: 892
X-RateLimit-Period: 3600
X-RateLimit-Remaining: 54999

Show we are allowed 55000 requests over an hour (3600 seconds)
Or, average 15 requests per second
1 request used and 892 sec (~15 minutes) from reset
Rate limiting

Requests are rate limited to prevent a single user from monopolising the resources.

**Retry-After: 40.0**
**X-RateLimit-Limit: 55000**
**X-RateLimit-Reset: 40**
**X-RateLimit-Period: 3600**
**X-RateLimit-Remaining: 0**

**Retry-After** header says we must wait 40 seconds before sending another request.

**429 HTTP will be returned**
Live example

- Loop 25 times fetching the ping endpoint
- Print the Status and RateLimit-Remaining header
- If we receive a Retry-After header, print that
Summary

We examined:

- How to query the Ensembl REST API
- Components of the endpoint documentation
- HTTP status codes used by the API
- Content types and retrieving data in different formats
- Rate limits on the API
- Using POST to batch your queries
Next in the series
See the full list of upcoming webinars at
http://www.ebi.ac.uk/training/webinars

Feedback
Tell us what you think
Help and documentation

Full documentation is available on the REST website

http://rest.ensembl.org

Email us helpdesk@ensembl.org
Ensembl public mailing lists dev@ensembl.org, announce@ensembl.org
Please cite us

http://www.ensembl.org/info/about/publications.html

Aken, B. et al
Ensembl 2017
Nucleic Acids Research
http://europepmc.org/articles/PMC5210575

DOI: https://doi.org/10.1093/nar/gkw1104
Ensembl Acknowledgements

The Entire Ensembl Team

Funding

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