Using RESTful web services

Under construction

Using REST web services

In this tutorial you will learn how to query CellBase web services using different technologies and programming languages. You are encouraged to visit **[[RESTful Web Services]]** documentation to understand and learn how CellBase web services have been designed and implemented.

There are several public CellBase RESTful servers you can use. In this tutorial we will use University if Cambridge CellBase host URL ***bioinfo.hpc.cam.ac.uk/cellbase/webservices/rest*** but you can change it to read data from any other host such as EMBL-EBI CellBase installation. You can also follow the [[Download and Installation]] and [[Load Data Models]] tutorials and install your own RESTful server, then you can use *localhost:8080/cellbase/webservices/rest*.

Technologies and programming languages:

- curl
- Python
- Java
- R
- Javascript

Using *curl*

curl is a command line tool and library for transferring data with URL syntax. It supports many different protocols including HTTP what allows to use it to query data. You can execute in the shell:

```
# Get information of Human genes located in Chromosome 3 between coordinates 55 and 100000
curl http://bioinfo.hpc.cam.ac.uk/cellbase/webservices/rest/v3/hsapiens/genomic/region/3:55-100000/gene

# Get all Human variants associated with beta-Thalassemia
curl http://bioinfo.hpc.cam.ac.uk/cellbase/webservices/rest/v3/hsapiens/genomic/variant/beta_Thalassemia/phenotype

# Get information for all Drosophila melanogaster chromosomes
curl http://bioinfo.hpc.cam.ac.uk/cellbase/webservices/rest/v3/dmelanogaster/genomic/chromosome/all
```

Using *Python*

Python is a general programming language that lets you work quickly and integrate systems more effectively. You can easily query RESTful web services using *urllib2* library. This snippet retrieves all the human genes in a particular region.

```
#!/usr/bin/python

import urllib2
import json
import zlib

# Prepare the call to the server
url = 'http://bioinfo.hpc.cam.ac.uk/cellbase/webservices/rest/v3/hsapiens/genomic/region/3:1104666-1166667/gene'
req = urllib2.Request(url)

# Inform to the server we accept gzip compression
req.add_header("Accept-Encoding", "gzip")

# Excute the call and read the result
response = urllib2.urlopen(req)
json_data = response.read()

# Uncompress the gzip result
data = zlib.decompress(json_data, 16+zlib.MAX_WBITS)
data = json.loads(data)

print data
```

Using Java

CellBase comes with a built-in Java client to make easier calling to web services:

```
import org.opencb.biodata.models.feature.Region;
import org.opencb.cellbase.core.client.CellBaseClient;
import org.opencb.cellbase.core.common.core.Gene;
import org.opencb.datastore.core.QueryOptions;
import org.opencb.datastore.core.QueryResponse;
import org.opencb.datastore.core.QueryResult;

public class CellBaseClientTest {
    public static void main(String[] args) throws URISyntaxException, IOException {
        CellBaseClient cellbaseClient = new CellbaseClient(
                "bioinfo.hpc.cam.ac.uk", 80, "cellbase/webservices/rest/", "v3", "hsapiens");

        QueryResponse<QueryResult<Gene>> response =
                cellBaseClient.getGene(CellBaseClient.Category.genomic, CellBaseClient.SubCategory.region,
                Arrays.asList(new Region("3", 1104666, 1166667)),
                new QueryOptions());

        System.out.println(response);
        return;
    }
}
```

You can also query RESTful web services using the Java ***Jersey*** library or any other RESTful client library:

```
import org.glassfish.jersey.client.ClientConfig;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.UriBuilder;

import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.ObjectReader;

public class CellBaseJerseyTest {

    private final ObjectMapper objectMapper = new ObjectMapper(new JsonFactory());
    private final Client client = ClientBuilder.newClient(new ClientConfig());

    public static void main(String[] args) throws URISyntaxException, IOException {
        Class<?> clazz = Gene.class;
        URI uri = new URI("http://bioinfo.hpc.cam.ac.uk/cellbase/webservices/rest/v3/hsapiens/genomic/region/3:
1104666-1166667/gene?of=json");
        UriBuilder uriBuilder = UriBuilder.fromUri(uri)

        Invocation.Builder request = client.target(uriBuilder).request();
        Response response = request.get();
        String responseStr = response.readEntity(String.class);

        ObjectReader responseReader = objectMapper.reader(objectMapper.getTypeFactory().constructParametricType(
                QueryResponse.class, objectMapper.getTypeFactory().constructParametricType(
                QueryResult.class, clazz))));

        QueryResponse<QueryResult<Gene>> response = responseReader.readValue(responseStr);

        System.out.println(response);
        return;
    }
}
```

Using R

R is a programming language and software environment widely used by statisticians and data analysts. You can easily query RESTful web services using *RCurl* and *rjson* libraries. This snippet retrieves all the human genes in a particular region.

```
library(RCurl)
library(rjson)

## Set URL
url <- 'http://bioinfo.hpc.cam.ac.uk/cellbase/webservices/rest/v3/hsapiens/genomic/region/3:1104666-1166667/gene?
exclude=transcripts'

## Retrieve data
response <- getURL(url)

## Transform JSON to list
json.data <- fromJSON(response)

## Count number of genes found
num.genes <- length(json.data$response[[1]]$result)
print( paste0("Number of genes found: ", num.genes))

## print genes
print("Genes:")
for (res in json.data$response[[1]]$result){
  print ( paste(res$name, res$id, res$biotype, sep = ", "))
}
```

Using JavaScript