GEO

Contents

1 Introduction

GEO[1] is a public data repository that is used to share files from different experiments. What we want to do with this database is to make it easy for users of our system to download files from that database directly to our server from a client. The interesting files for the users are the .sra files, these will be unpacked to .fastq files, this can be done with the sra-toolkit application. This application can be downloaded from here[2]. The API for the communication described below can be found under the GEO section of the API section in the technical documentation.

2 How will it work

The download of files will work like this:

1. The user uses the browser to search for different experiments on GEOs website.
2. When the user finds an interesting experiment, the user can enter the experiments GSE id number (this might look something like this GSE47236) in the client and press "search".
3. The server will then search for the id in GEO and download what is called "series matrix file". This file contains information about the experiment and where the URLs to the .sra that might be interesting for the user.
4. The server will then parse this series matrix file and extract as much information as possible.
5. The server will then send a json with all information that was successfully extracted to the client that made the search.
6. The information will be presented for the user in the client, and the user can then choose which files to be downloaded.
7. The client will then send a request to the server asking it to download the selected files.

[1] http://www.ncbi.nlm.nih.gov/geo/
[2] http://eutils.ncbi.nih.gov/Traces/sra/?view=software

3 What is done

A lot of work is already done on the server side, although there are some things left to do, and nothing have yet been integrated with the server.
What is done:

1. Receiving a request to search for a GSE number. (Server)
2. Downloading the series matrix file for the given GSE number. (Server)
3. Parse the series matrix file. (Server)
4. Create a json from the matrix file, and send it to the client. (Server)
5. A .php script that will download a file from a URL to a filepath. (Server)

What is not done (See 5 and 6 for more details):

1. Implement the GEO functionality on the server. (Server)
2. Converting the .php script to java, the process of this has started, but is not finished. (Server)
3. Send a search request from the client. (Client)
4. Receive the json containing the searchresult. (Client)
5. Show the information to the user in a usable way. (Client)
6. Send a request to the server to download the selected .sra files. (Client)

4 How to implement on Server side

To implement GEO for the server a few files must be added, and some code must be added to existing files. The files that are needed:

- GEOAccessor.java
- GEOCommand.java
- GEODownloadFileCommand.java (not known how to implement)
- GEOFileDownloader.java
- GEOFileTuple.java

- GetGEOResponse.java
- TXTParser.java

Those files (as of now) should be placed in a package named "geo", the files can be moved, but make sure the imports are to the right package. The code that needs to be added in existing files are:

CommandFactory.java
Just add a new method:
```
public Command createSearchGEO(String json) {
    return new GEOCommand(json);
}
```

CommandHandler.java
In the big if-else in the createComment method, add:
```
else if (cmdt == CommandType.GET_GEO_ID) {
    newCommand = cmdFactory.createSearchGEO(json);
}
```

CommandType.java
Add a new enum:
```
GET_GEO_ID;
```

Doorman.java
Add this to the constructor, it should be clear where:
```
httpServer.createContext("/geo", createHandler());
```

And in the method createHandler(), in the GET switch add another case:
```
case "/geo":
    exchange(exchange, CommandType.GET_GEO_ID);
    break;
```

The server should now be ready to receive a request to search for a GSE number and then respond with a json with information about what files are avalible to download.
The implementation of the `GEODownloadFileCommand` should be similar to the above, although this has not been tested.

5 What to do now (Server)

Right now, there is a class in `GEOFileDownloader.java` that have 2 implemented methods. One is `getSRAFromDir(String url)`, in the series matrix file, all the URLs to the .sra files point to a directory, and this method will navigate trough the directory and find the full URL to the .sra file and return it.

The other is `downloadFile(String url, String tempFileLoc)`, this method will run a .php script that is located on the server (`download_from_geo.php`), this script is located with the other .php scripts `download.php` and `upload.php`. The scrip `download_from_geo.php` downloads a .sra file from the URL and saves it to the path tempFileLoc, when the file is downloaded the script will use SRA-Toolkit to extract the .sra file to a .fastq file.

The `downloadFile(String url, String tempFileLoc)` method is being replaced with the method `downloadFileDirectly(String url, String tempFileLoc)`. Although this new method is not yet finished, this new method will replace the .php script that is currently handling the download from GEO. The reason for the conversion from PHP to Java is that it will be easier to handle errors and to keep everything in the same place.

6 What to do now (Client)

Since nothing on the Client has been implemented, there is a lot to do. The first thing to do is a simple searchfield where the user can enter the GSE number. Then the client must send this number to the server. The client will then receive a json containing a lot of information, the users stated that they want to see all information from the series matrix file, all this information is sent in the json. But since it is quit a lot of information, it is up to you to decide what information is necessary to view. But the most important information is ID_REF and sampleSupplementaryFile, the first one is the name of the file, the second one is the URL to the .sra file. These two should be sent back to the server so it can be downloaded.

7 Classdescription

This section is a description of the classes that concerns the GEO databse.

7.1 GEOAccessor.java

This class contains a method `getMatrixFileURL(String id)` that searches for the GSE number int the GEO database. Then when the method `Stirng matrixFileURL, String tempFileLoc` is called the series matrix file for that given GSE number will be downloaded. Then to unpack the series matrix file call `ginzipFile(String tempZipFileLoc, String tempFileLoc)`, this method will unzip the matrix series file.

7.2 GEOCommand.java

This class uses the methods in `GEOAccessor` to download and unpack the series matrix file. This class extends the Command class and is used in the server when a series matrix file should be downloaded. The `execute()` method will return a response containing information from the parsed series matrix file.

7.3 GEODownloadFileCommand.java

This class extends `Command` and is the class that will be called when the server wants to download a file from GEO. The `execute()` method in this class will download the .sra file from the GEO database.

7.4 GEOFileDownloader.java

This class have a method named `getSRAFromDir(String url)`, this method takes a string in the form of an URL, this URL is parsed from the series matrix file, and that URL is only a URL to a directory, this method will search the directory for .sra files and return a complete URL to a .sra file so it can be downloaded. This class also have a method named `downloadFile(String url, String tempFileLoc)`, this method will run a .php script that is located on the server (`download_from_geo.php`). There is also a method that is not yet finished named `downloadFileDirectly(String url, String tempFileLoc)` this method is supposed to download the .sra file and convert it without using the .php script.

7.5 GEOFileTuple.java

This class only holds information about experiments.

7.6 GetGeoResponse.java

The constructor int this class takes an ArrayList with `GEOFileTuple` objects and creates a json string from the objects. The list with `GEOFileTuple`s is returned from the `TXTParser`. Then the `GEOFileTuple` objects will be coverted to a json string that can be retrieved with the method `getBody()`.

7.7 TXTParser.java

This class is basically a method (`readFile(String filePath)`) with a huge if-else. This method will parse the series matrix file and create an ArrayList containing `GEOFileTuple` objects. The first object in the list is a object containing all the information that is mutual for the whole experiment. The part of the series matrix file that is mutual is the rows that begin with "Series". All the other objects in the list are `GEOFileTuple` that contains information about individual experiments. The information that is individual starts with "Sample". There are also some private methods that is used to format the strings that are being parsed.