Zendesk

Description

The project is an initial level command-line interface to connect to the zendesk project. The java code has been written in a very easy and simple language for understandability. The few aspects of the code are it performs Security by passing an API token and encrypting it using Base64 and sending it to the Zendesk API. Here, if the API token and email ID does not match, the code throws an error while using the switch Case.

**** The code takes the input of API Token. This is implemented for complete security reasons only. Only a specified user can access the particular account information. We can give it directly as well in the code without taking input from the user. ****

Installation

There are two files - Zendesk.java and ZendeskFunctions.java. You can run them in Eclipse or any other IDE of your choice

1. *Zendesk*

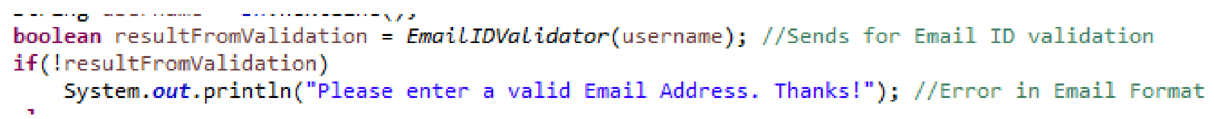
This is the main file to be executed. It calls all functions from the switch case to be executed.

2. *ZendeskFunctions*

This file contains all the important functions that are necessary for successful execution.

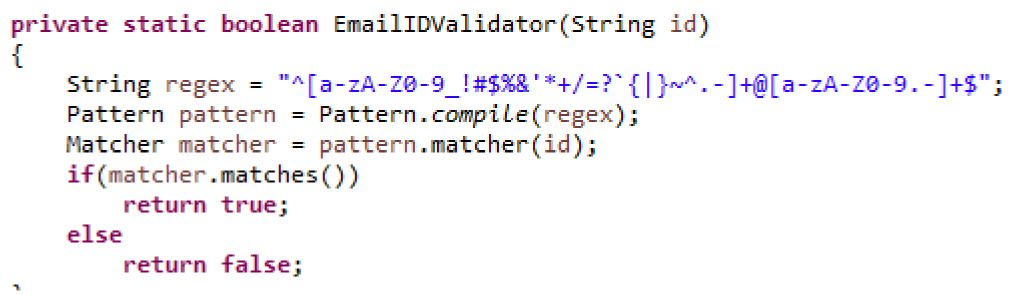
You also need to add an external library and add a classpath to your project for a file called "java-son.jar" to process JSON data.

Security



```
boolean resultFromValidation = EmailIDValidator(username); //Sends for Email ID validation
if(!resultFromValidation)
    System.out.println("Please enter a valid Email Address. Thanks!"); //Error in Email Format
```

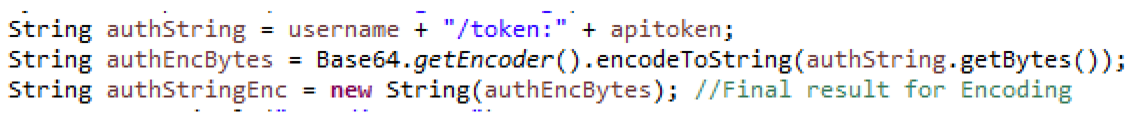
Step1: The code asks for the email Id. This is the one that you have used on your Zendesk API. The prior part here is Email ID verification. The use of regex is been done to check if a valid email address has been passed. If yes, then the program runs ahead or else quits.



```java
private static boolean EmailIDValidator(String id)
{
    String regex = "^[a-zA-Z0-9_!#$%&'*+/=?`{|}~^.-]+@[a-zA-Z0-9.-]+$";
    Pattern pattern = Pattern.compile(regex);
    Matcher matcher = pattern.matcher(id);
    if(matcher.matches())
        return true;
    else
        return false;
```

Step2: The code asks for an API token. This is the one that you have saved and Enable on the Zendesk API token. Make sure you enable both the options on Zendesk API

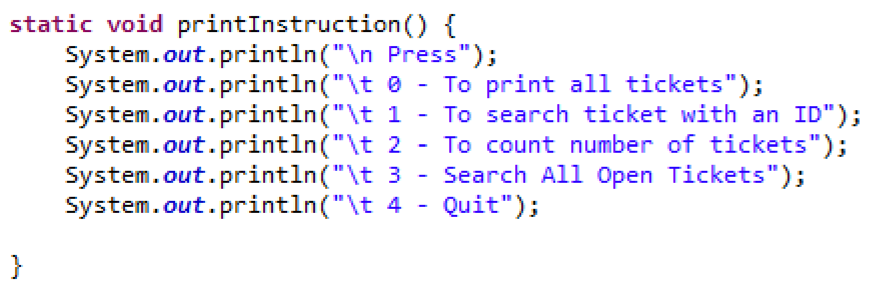
Step3: There performs an internal Base64 conversion of your API token and Email Id that acts as the base for your authentication header of HTTPRequest



```java
String authString = username + "/token:" + apitoken;
String authEncBytes = Base64.getEncoder().encodeToString(authString.getBytes());
String authStringEnc = new String(authEncBytes); //Final result for Encoding
```

Step4: It asks for the domain name. The importance of asking this is a user can access tickets of its account. Everything is being stored and combined in different strings so that the user has no control over any links of the server. Everything has been automated making it easy for the user to handle the tickets.

Switch Case functions



```java
static void printInstruction() {
    System.out.println("\n Press");
    System.out.println("\t 0 - To print all tickets");
    System.out.println("\t 1 - To search ticket with an ID");
    System.out.println("\t 2 - To count number of tickets");
    System.out.println("\t 3 - Search All Open Tickets");
    System.out.println("\t 4 - Quit");

}
```

- 0 : It is used to send all the tickets on your respective account. We have tried to use pagination so that a max of 25 tickets would be displayed

- 1 : It is used to search tickets with the specified ID. All the details of the specified ID tickets would be displayed

- 2 : It is used to give an overall count of the number of tickets in the account

- 3 : This options displays all the open tickets.

- 4 : This exits the code and breaks the while loop

General Communication Features

Variables here have been used as to what they perform in the scripts. This makes the nature of code understandable and easy to read.

- username : Its the email id of the user

- apitoken : The API token that has been stored on database

- authString : Combination of username and APi token

- authEncBytes : Byte format of 64bit encoded string

- authStringEnc : Encoded String final result

- domain : domain of your zendesk account. They are the words before ".zendesk.com"

- url: Readymade URL formed by combining domain and zendesk API

- choice: Switch case input

- regex: Stores Email Id validator pattern

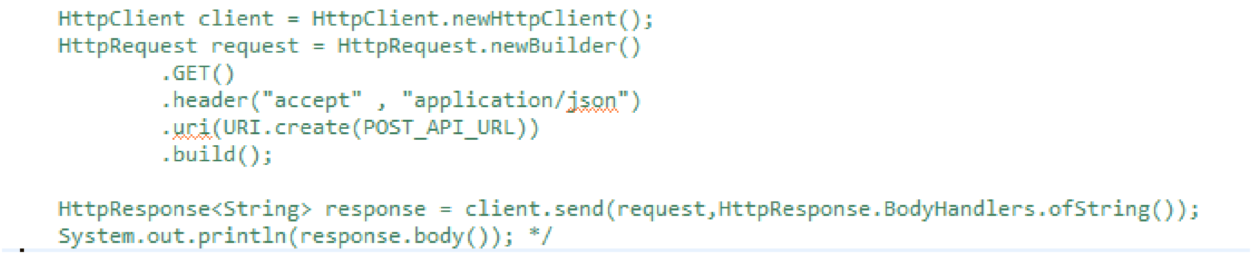
- pattern: Stores compilation of regex pattern

- matcher: Stores output that has been given out after matching with the input id

- EncodeURL: Every function has its URL that has been specific to their usage

Extensibility of Project

This was the previous code that was used in the initial development

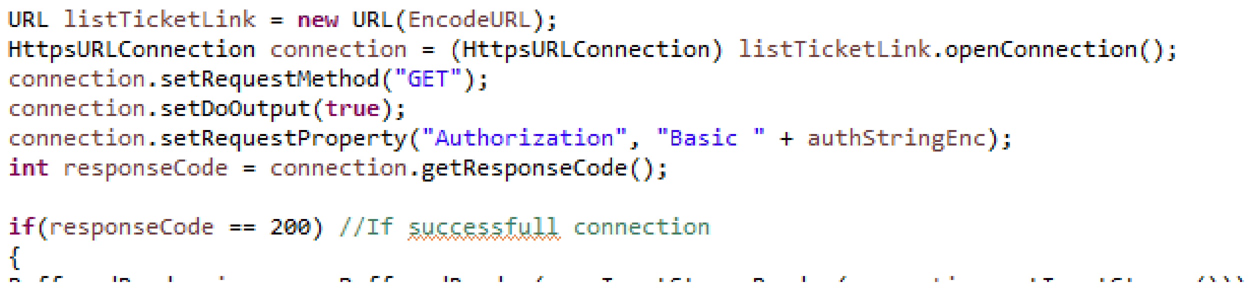


```
HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
        .GET()
        .header("accept" , "application/json")
        .uri(URI.create(POST_API_URL))
        .build();

HttpResponse<String> response = client.send(request,HttpResponse.BodyHandlers.ofString());
System.out.println(response.body()); */
```

The drawbacks were no handling of Exceptions and errors. There was no functionality to get the response code from the server making it difficult to understand where the problem lies.

The updated code extends the project with much wider functions :

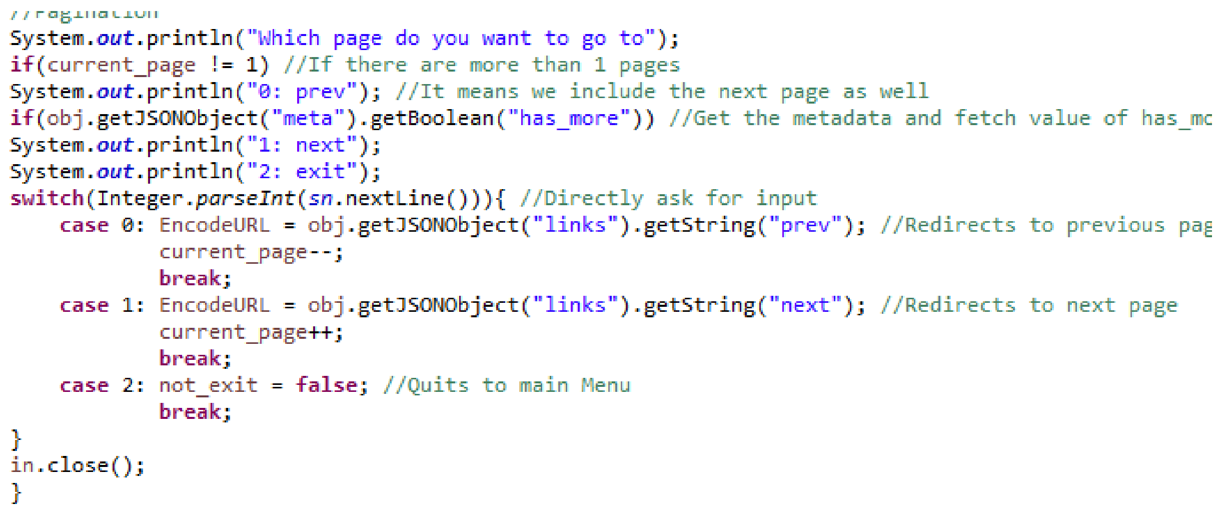


```
URL listTicketLink = new URL(EncodeURL);
HttpsURLConnection connection = (HttpsURLConnection) listTicketLink.openConnection();
connection.setRequestMethod("GET");
connection.setDoOutput(true);
connection.setRequestProperty("Authorization", "Basic " + authStringEnc);
int responseCode = connection.getResponseCode();

if(responseCode == 200) //If successfull connection
{
[illegible]
```

Here, we use the most reliable and updated Http class called HttpsURLConnection. The reason being it comes to various inbuilt classes and functions to be used. If we use HttpRequest and HttpClient, it would be much harder to get the response code and handle the error. We instead use this one where we could get the response code from the server which is 200 or 400 or any other simply by calling .getResponseCode(). This makes it easier to display the error and IOExceptions are handled easily.

Understanding Pagination

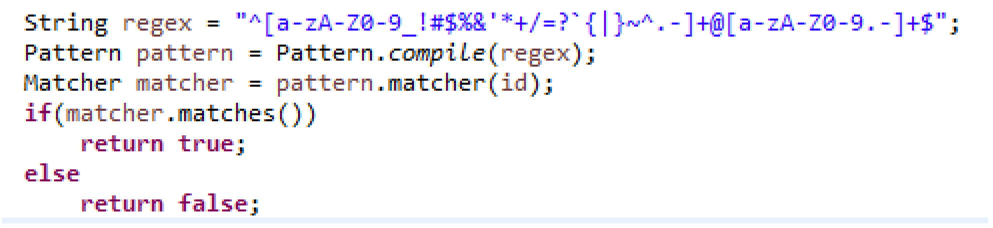


```
//Pagination
System.out.println("Which page do you want to go to");
if(current_page != 1) //If there are more than 1 pages
System.out.println("0: prev"); //It means we include the next page as well
if(obj.getJSONObject("meta").getBoolean("has_more")) //Get the metadata and fetch value of has_mc
System.out.println("1: next");
System.out.println("2: exit");
switch(Integer.parseInt(sn.nextLine())){ //Directly ask for input
    case 0: EncodeURL = obj.getJSONObject("links").getString("prev"); //Redirects to previous pag
            current_page--;
            break;
    case 1: EncodeURL = obj.getJSONObject("links").getString("next"); //Redirects to next page
            current_page++;
            break;
    case 2: not_exit = false; //Quits to main Menu
            break;
}
in.close();
}
```

Pagination has been implemented using a switch case. At first, the code checks if the page is not 1 which means it's greater than 1. So Zendesk provides meta which stores the Boolean value called "has_more" which stores if there are any more pages. If there are, then only it will proceed with the Switch case else terminates. The JSON object contains an option called links that stores little information. We extract the value of previous or next depending upon the user input.

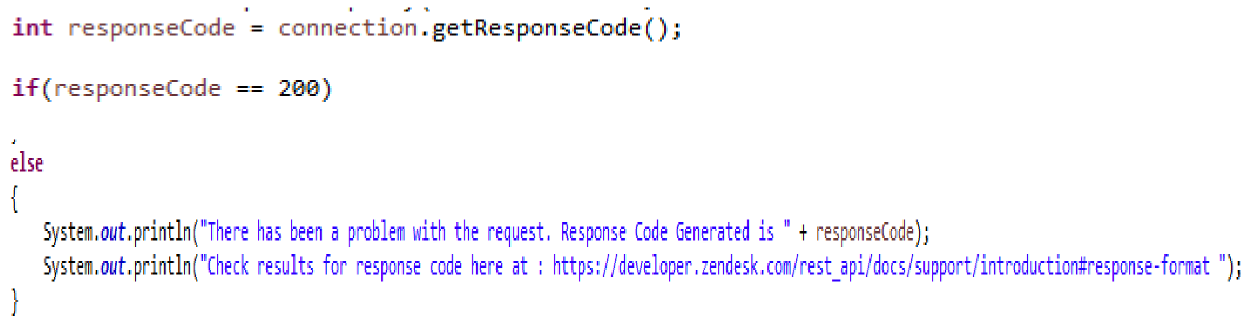
Error Handling Code

The code has implemented different mechanism for error handling. There has been email ID validator as per the RFC 5322 regex.



```java
String regex = "^[a-zA-Z0-9_!#$%&'*+/=?`{|}~^.-]+@[a-zA-Z0-9.-]+$";
Pattern pattern = Pattern.compile(regex);
Matcher matcher = pattern.matcher(id);
if(matcher.matches())
    return true;
else
    return false;
```

Along with Email ID validator , response code handling has been done which prints the link that helps and supports the corresponding response code



```java
int responseCode = connection.getResponseCode();

if(responseCode == 200)

else
{
    System.out.println("There has been a problem with the request. Response Code Generated is " + responseCode);
    System.out.println("Check results for response code here at : https://developer.zendesk.com/rest_api/docs/support/introduction#response-format ");
}
```