2018 Citi Financial Innovation Application Competition

API Incocation Report

SpecialPlan
砺金专项计划

Title ： Li Jin--A REITs platform for securitization of housing lease assets

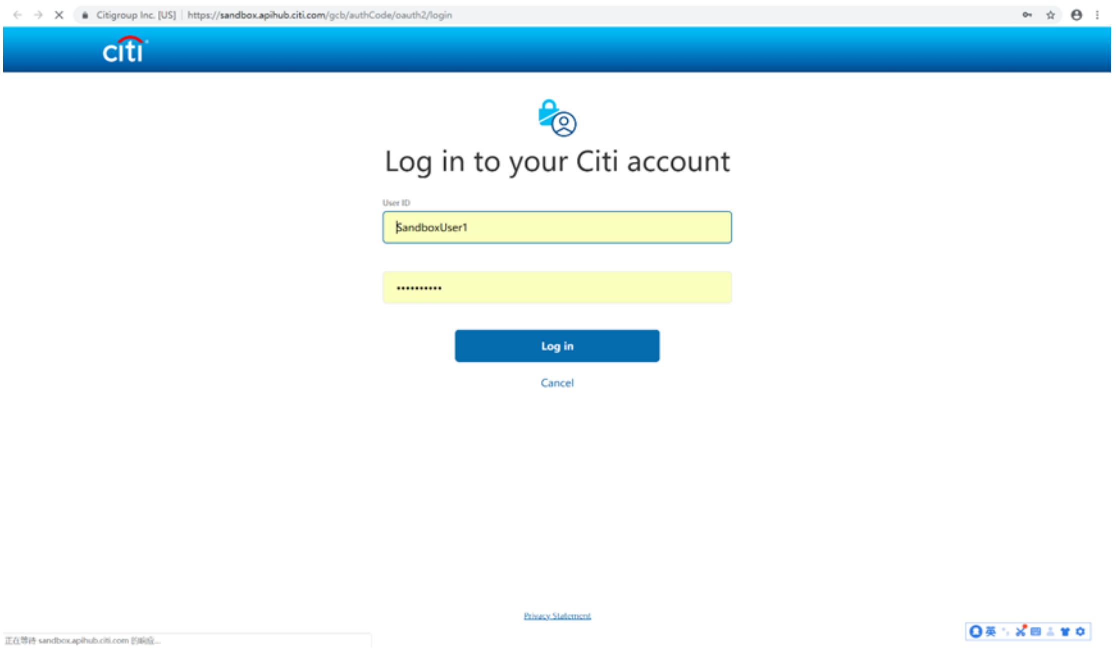
Captain： Chu Tianshuo

Tutor： Gao Ming, Sui Cong

School： Dongbei University of Finance and Economics

Contents

1. Part One: Project Exhibition



Citigroup Inc. [US] | https://sandbox.apihub.citi.com/gcb/authCode/oauth2/login

Log in to your Citi account

User ID

SandboxUser1

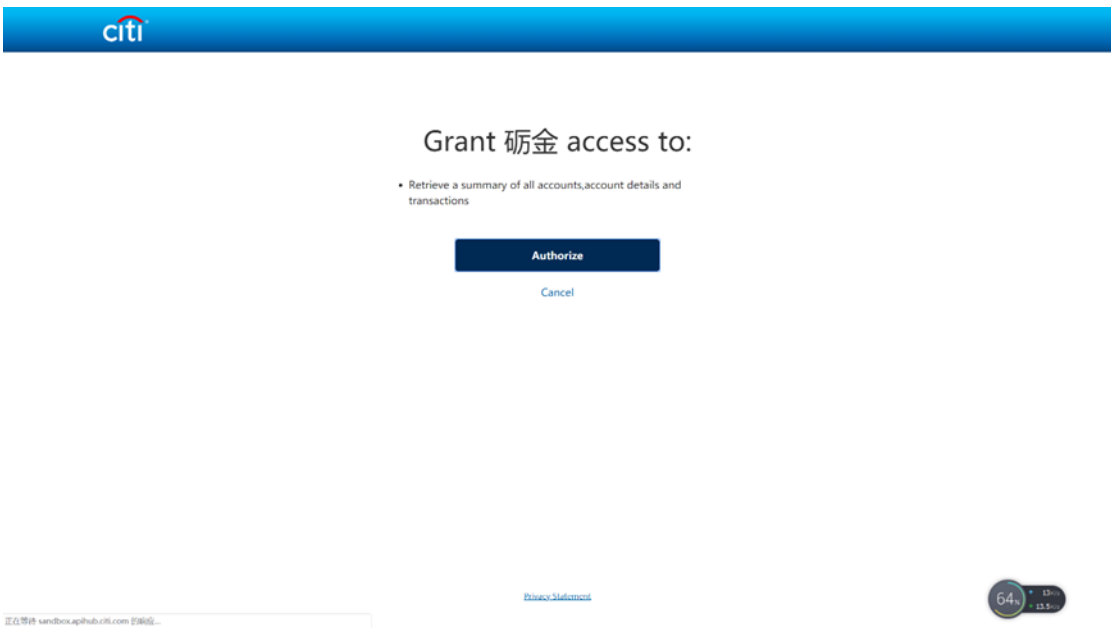
••••••••••

Log in

Cancel

Privacy Statement

正在等待 sandbox.apihub.citi.com 的响应...



Grant 砺金 access to:

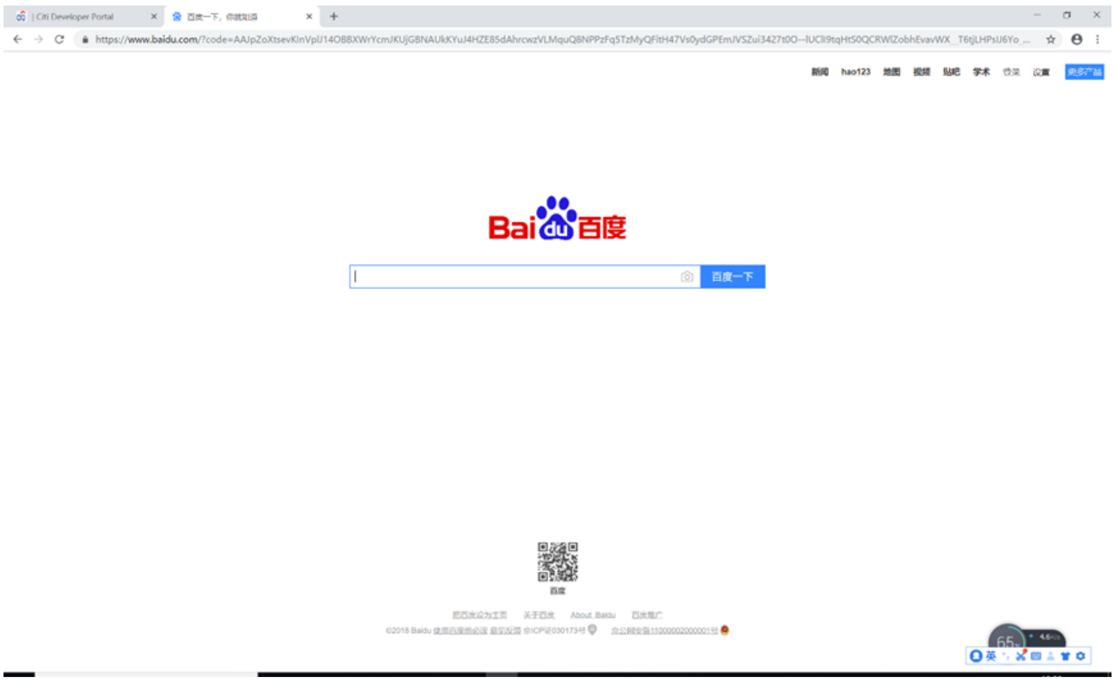
- Retrieve a summary of all accounts,account details and transactions

Authorize

Cancel

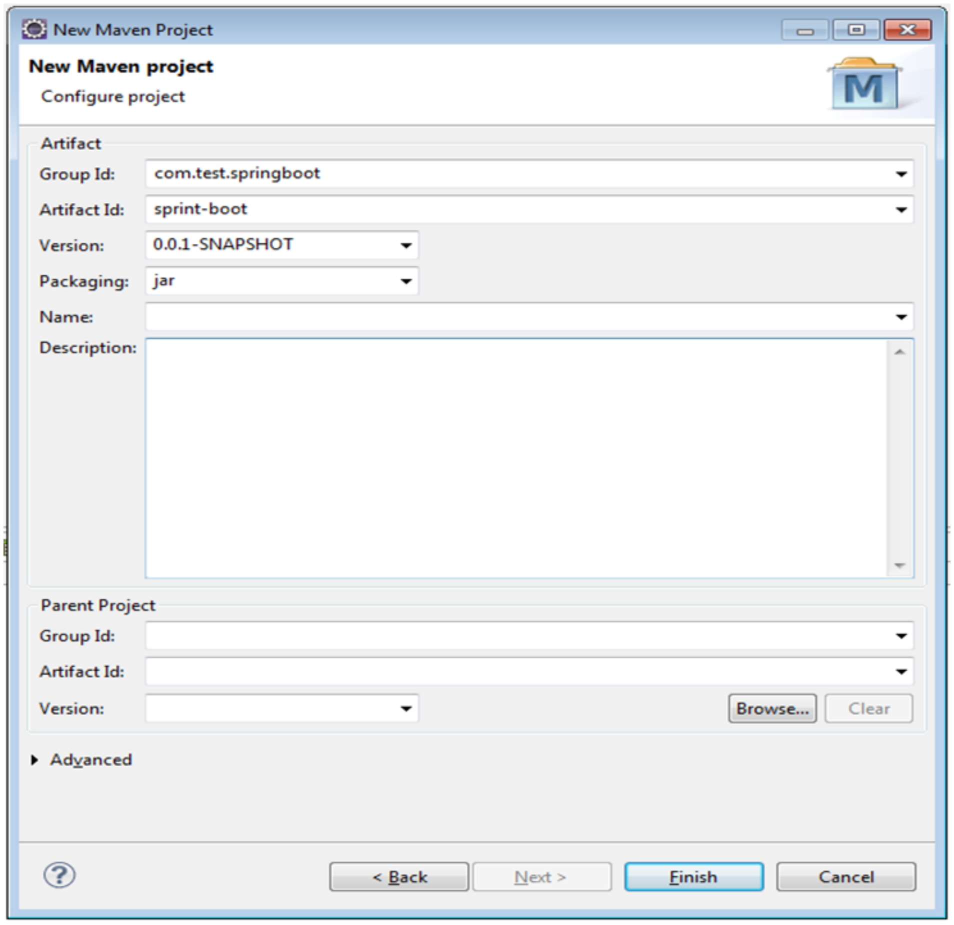
Privacy Statement

正在等待 sandbox.apihub.citi.com 的响应...



2.Part Two: Specific Steps

（1）Establish Maven Project.



New Maven Project

New Maven project

Configure project

Artifact

Group Id: com.test.springboot

Artifact Id: sprint-boot

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name:

Description:

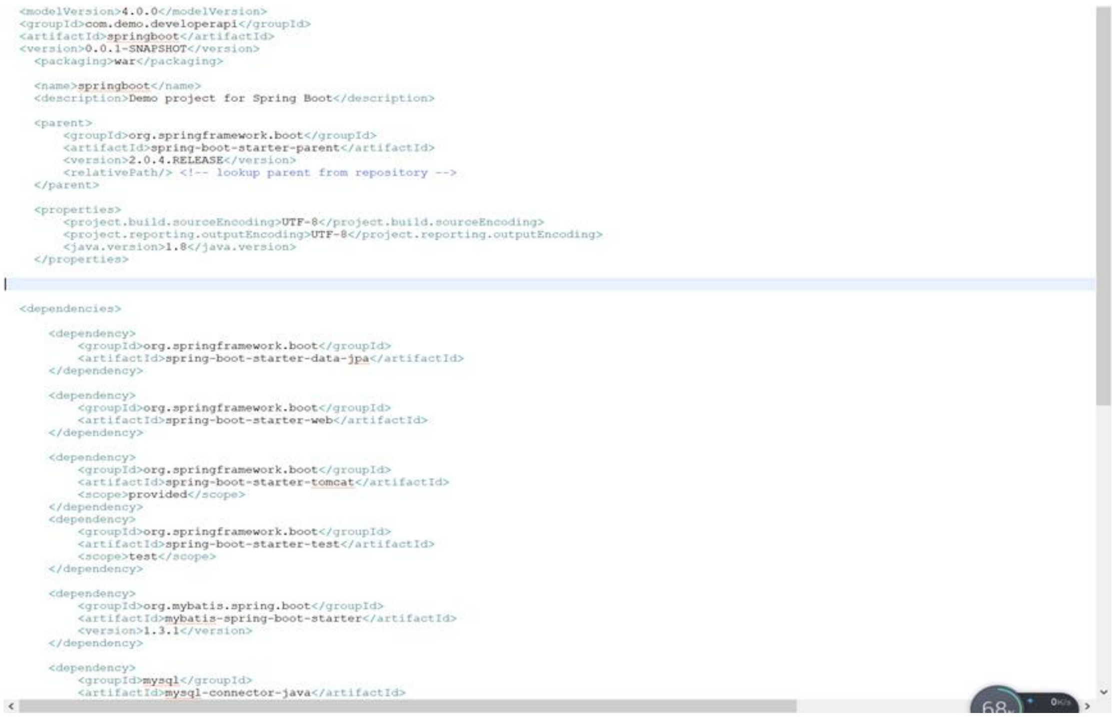
Parent Project

Group Id:

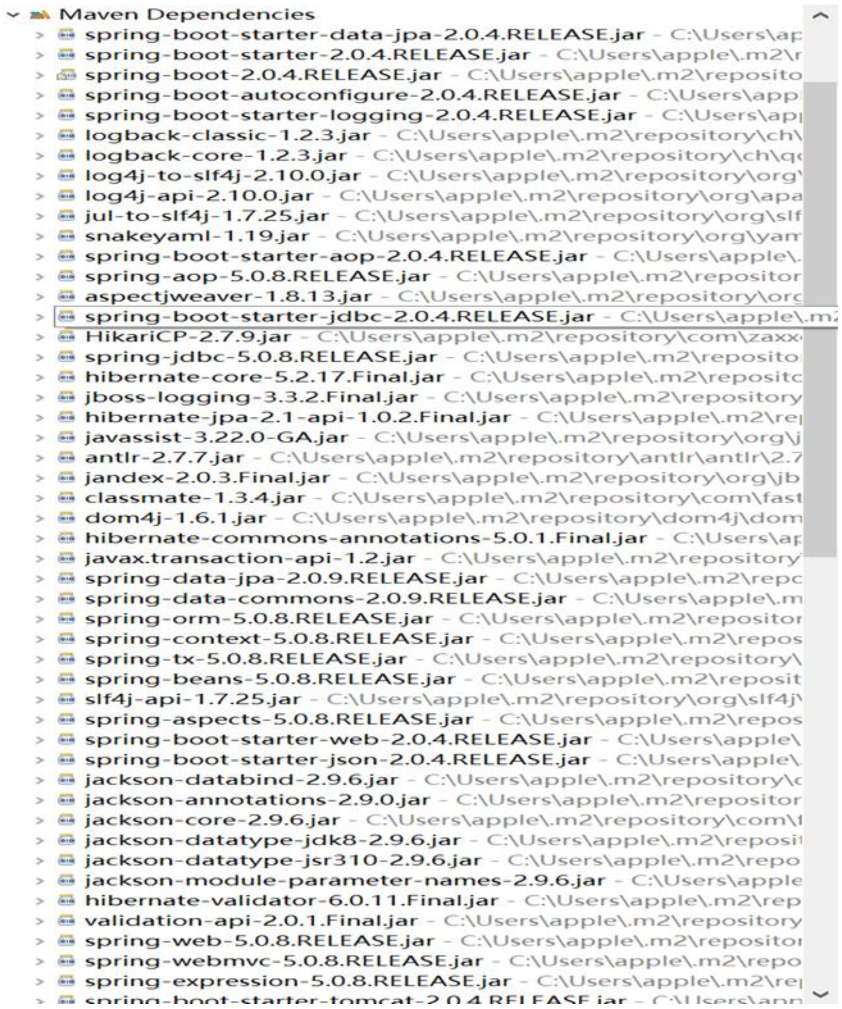
Artifact Id:

Version:

（2）Edit the pom.xml file and add the required jar package.

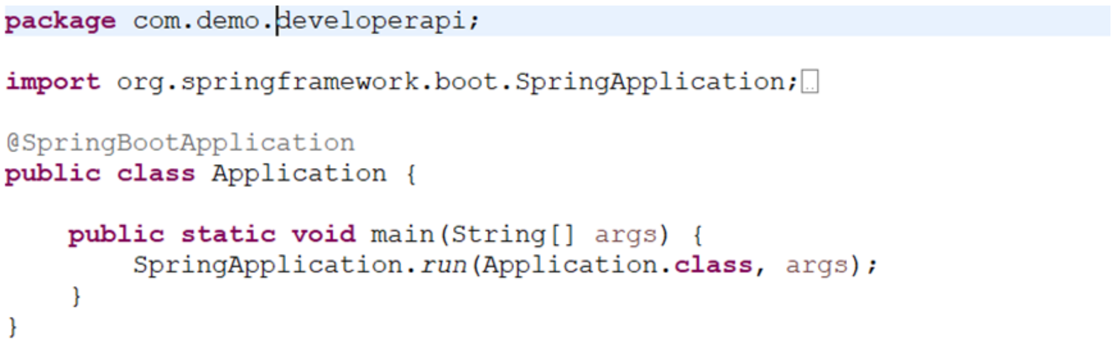


```xml
<modelVersion>4.0.0</modelVersion>
<groupId>com.demo.developerapi</groupId>
<artifactId>springboot</artifactId>
<version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>springboot</name>
  <description>Demo project for Spring Boot</description>

  <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>2.0.4.RELEASE</version>
      <relativePath/> <!-- lookup parent from repository -->
  </parent>

  <properties>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
      <java.version>1.8</java.version>
  </properties>


<dependencies>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>org.mybatis.spring.boot</groupId>
        <artifactId>mybatis-spring-boot-starter</artifactId>
        <version>1.3.1</version>
    </dependency>

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
```



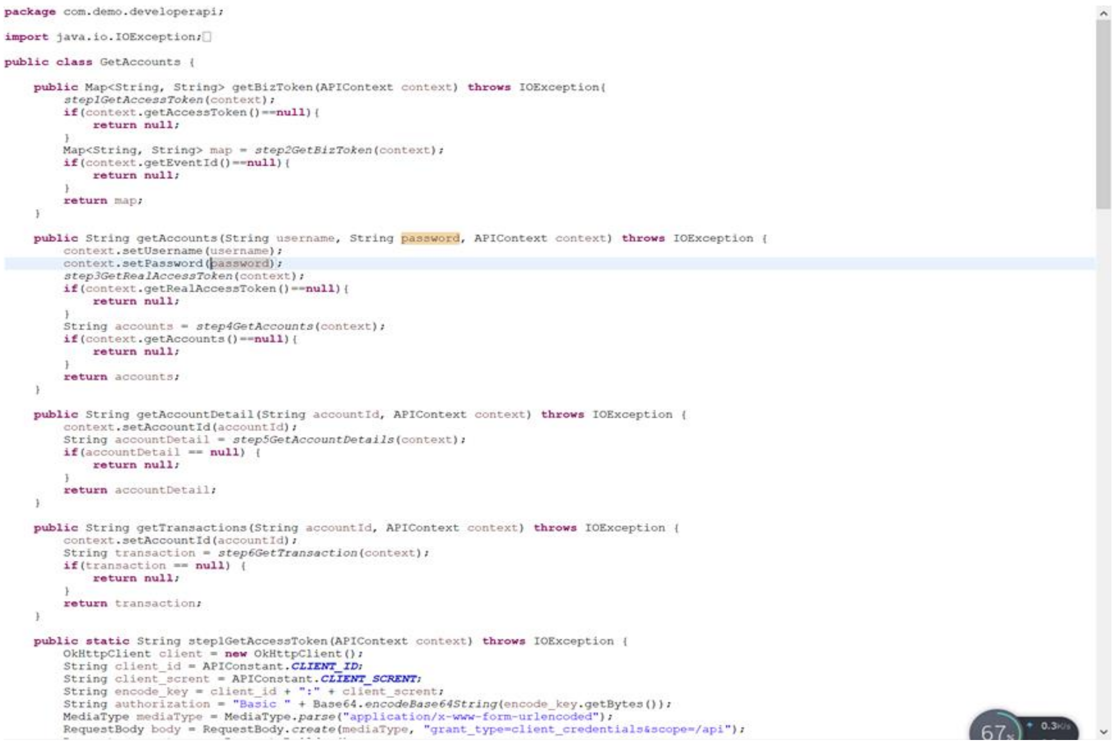
```
Maven Dependencies
  spring-boot-starter-data-jpa-2.0.4.RELEASE.jar - C:\Users\ap
  spring-boot-starter-2.0.4.RELEASE.jar - C:\Users\apple\.m2\r
  spring-boot-2.0.4.RELEASE.jar - C:\Users\apple\.m2\reposito
  spring-boot-autoconfigure-2.0.4.RELEASE.jar - C:\Users\app
  spring-boot-starter-logging-2.0.4.RELEASE.jar - C:\Users\ap
  logback-classic-1.2.3.jar - C:\Users\apple\.m2\repository\ch\
  logback-core-1.2.3.jar - C:\Users\apple\.m2\repository\ch\q
  log4j-to-slf4j-2.10.0.jar - C:\Users\apple\.m2\repository\org\
  log4j-api-2.10.0.jar - C:\Users\apple\.m2\repository\org\apa
  jul-to-slf4j-1.7.25.jar - C:\Users\apple\.m2\repository\org\slf
  snakeyaml-1.19.jar - C:\Users\apple\.m2\repository\org\yam
  spring-boot-starter-aop-2.0.4.RELEASE.jar - C:\Users\apple\.
  spring-aop-5.0.8.RELEASE.jar - C:\Users\apple\.m2\repositor
  aspectjweaver-1.8.13.jar - C:\Users\apple\.m2\repository\org
  spring-boot-starter-jdbc-2.0.4.RELEASE.jar - C:\Users\apple\.m2
  HikariCP-2.7.9.jar - C:\Users\apple\.m2\repository\com\zaxx
  spring-jdbc-5.0.8.RELEASE.jar - C:\Users\apple\.m2\reposito
  hibernate-core-5.2.17.Final.jar - C:\Users\apple\.m2\reposito
  jboss-logging-3.3.2.Final.jar - C:\Users\apple\.m2\repository
  hibernate-jpa-2.1-api-1.0.2.Final.jar - C:\Users\apple\.m2\re
  javassist-3.22.0-GA.jar - C:\Users\apple\.m2\repository\org\j
  antlr-2.7.7.jar - C:\Users\apple\.m2\repository\antlr\antlr\2.7
  jandex-2.0.3.Final.jar - C:\Users\apple\.m2\repository\org\jb
  classmate-1.3.4.jar - C:\Users\apple\.m2\repository\com\fast
  dom4j-1.6.1.jar - C:\Users\apple\.m2\repository\dom4j\dom
  hibernate-commons-annotations-5.0.1.Final.jar - C:\Users\ap
  javax.transaction-api-1.2.jar - C:\Users\apple\.m2\repository
  spring-data-jpa-2.0.9.RELEASE.jar - C:\Users\apple\.m2\repo
  spring-data-commons-2.0.9.RELEASE.jar - C:\Users\apple\.m
  spring-orm-5.0.8.RELEASE.jar - C:\Users\apple\.m2\repositor
  spring-context-5.0.8.RELEASE.jar - C:\Users\apple\.m2\repos
  spring-tx-5.0.8.RELEASE.jar - C:\Users\apple\.m2\repository\
  spring-beans-5.0.8.RELEASE.jar - C:\Users\apple\.m2\reposit
  slf4j-api-1.7.25.jar - C:\Users\apple\.m2\repository\org\slf4j\
  spring-aspects-5.0.8.RELEASE.jar - C:\Users\apple\.m2\repos
  spring-boot-starter-web-2.0.4.RELEASE.jar - C:\Users\apple\
  spring-boot-starter-json-2.0.4.RELEASE.jar - C:\Users\apple\.
  jackson-databind-2.9.6.jar - C:\Users\apple\.m2\repository\c
  jackson-annotations-2.9.0.jar - C:\Users\apple\.m2\repositor
  jackson-core-2.9.6.jar - C:\Users\apple\.m2\repository\com\f
  jackson-datatype-jdk8-2.9.6.jar - C:\Users\apple\.m2\reposi
  jackson-datatype-jsr310-2.9.6.jar - C:\Users\apple\.m2\repo
  jackson-module-parameter-names-2.9.6.jar - C:\Users\apple
  hibernate-validator-6.0.11.Final.jar - C:\Users\apple\.m2\rep
  validation-api-2.0.1.Final.jar - C:\Users\apple\.m2\repository
  spring-web-5.0.8.RELEASE.jar - C:\Users\apple\.m2\repositor
  spring-webmvc-5.0.8.RELEASE.jar - C:\Users\apple\.m2\repo
  spring-expression-5.0.8.RELEASE.jar - C:\Users\apple\.m2\re
  spring-boot-starter-tomcat-2.0.4.RELEASE.jar - C:\Users\app
```

（3）Create Application.java to start the master program.



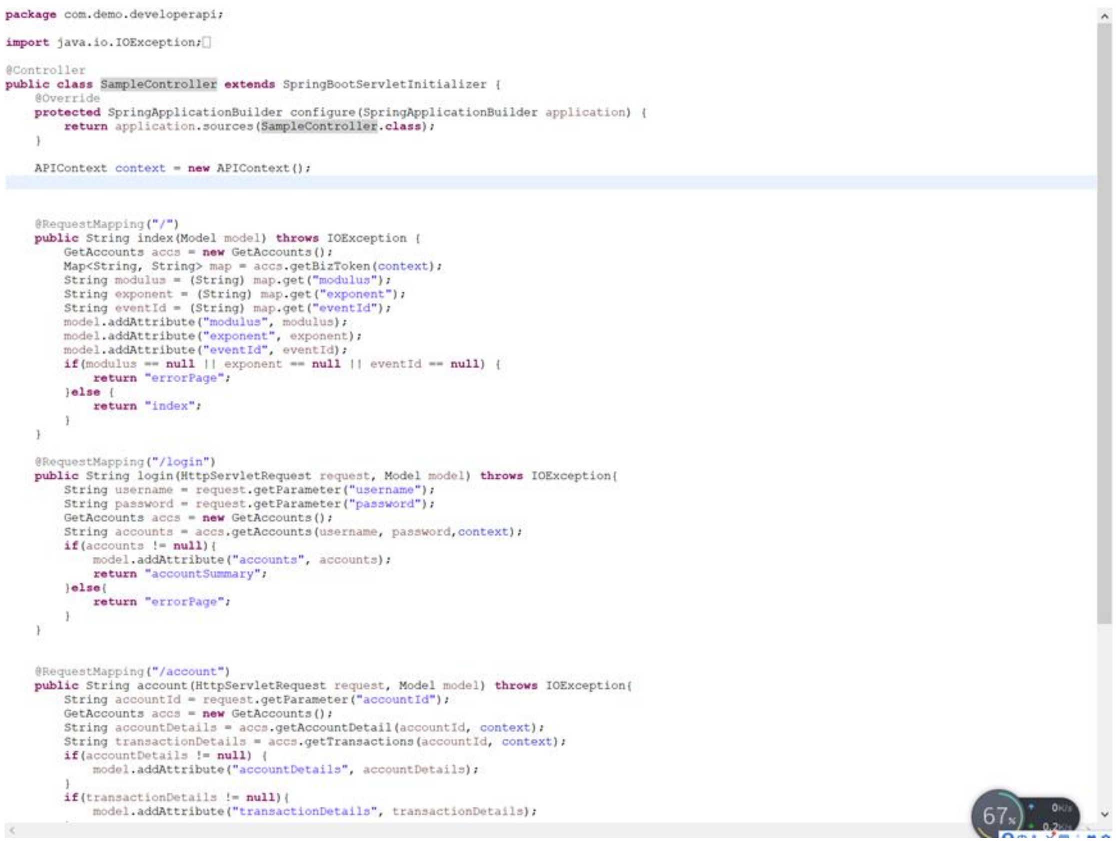
```java
package com.demo.developerapi;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

（4）Create GetAccounts.java and write the API function to be implemented.



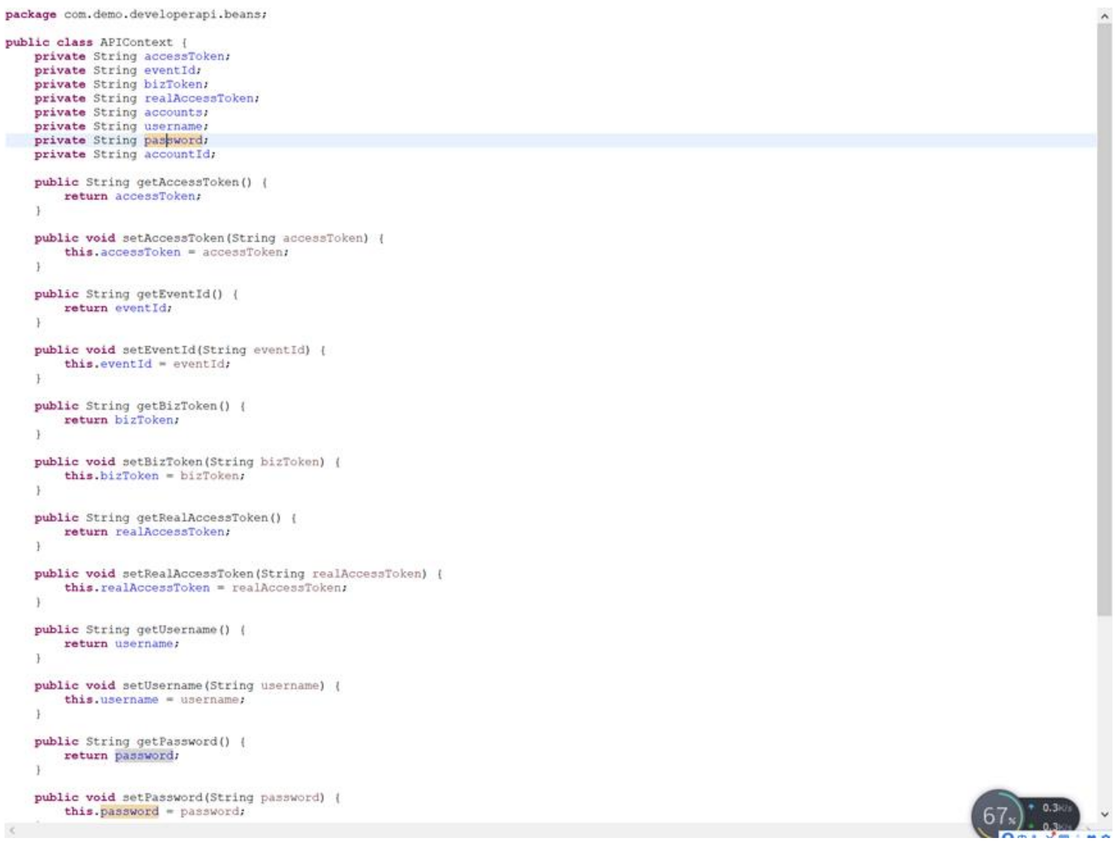
```java
package com.demo.developerapi;

import java.io.IOException;

public class GetAccounts {

    public Map<String, String> getBizToken(APIContext context) throws IOException{
        step1GetAccessToken(context);
        if(context.getAccessToken()==null){
            return null;
        }
        Map<String, String> map = step2GetBizToken(context);
        if(context.getEventId()==null){
            return null;
        }
        return map;
    }

    public String getAccounts(String username, String password, APIContext context) throws IOException {
        context.setUsername(username);
        context.setPassword(password);
        step3GetRealAccessToken(context);
        if(context.getRealAccessToken()==null){
            return null;
        }
        String accounts = step4GetAccounts(context);
        if(context.getAccounts()==null){
            return null;
        }
        return accounts;
    }

    public String getAccountDetail(String accountId, APIContext context) throws IOException {
        context.setAccountId(accountId);
        String accountDetail = step5GetAccountDetails(context);
        if(accountDetail == null) {
            return null;
        }
        return accountDetail;
    }

    public String getTransactions(String accountId, APIContext context) throws IOException {
        context.setAccountId(accountId);
        String transaction = step6GetTransaction(context);
        if(transaction == null) {
            return null;
        }
        return transaction;
    }

    public static String step1GetAccessToken(APIContext context) throws IOException {
        OkHttpClient client = new OkHttpClient();
        String client_id = APIConstant.CLIENT_ID;
        String client_scrent = APIConstant.CLIENT_SCRENT;
        String encode_key = client_id + ":" + client_scrent;
        String authorization = "Basic " + Base64.encodeBase64String(encode_key.getBytes());
        MediaType mediaType = MediaType.parse("application/x-www-form-urlencoded");
        RequestBody body = RequestBody.create(mediaType, "grant_type=client_credentials&scope=/api");
```

（5）Create SampleController.java and write the port to be invoked.



```java
package com.demo.developerapi;

import java.io.IOException;

@Controller
public class SampleController extends SpringBootServletInitializer {
    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(SampleController.class);
    }

    APIContext context = new APIContext();



    @RequestMapping("/")
    public String index(Model model) throws IOException {
        GetAccounts accs = new GetAccounts();
        Map<String, String> map = accs.getBizToken(context);
        String modulus = (String) map.get("modulus");
        String exponent = (String) map.get("exponent");
        String eventId = (String) map.get("eventId");
        model.addAttribute("modulus", modulus);
        model.addAttribute("exponent", exponent);
        model.addAttribute("eventId", eventId);
        if(modulus == null || exponent == null || eventId == null) {
            return "errorPage";
        }else {
            return "index";
        }
    }

    @RequestMapping("/login")
    public String login(HttpServletRequest request, Model model) throws IOException{
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        GetAccounts accs = new GetAccounts();
        String accounts = accs.getAccounts(username, password,context);
        if(accounts != null){
            model.addAttribute("accounts", accounts);
            return "accountSummary";
        }else{
            return "errorPage";
        }
    }


    @RequestMapping("/account")
    public String account(HttpServletRequest request, Model model) throws IOException{
        String accountId = request.getParameter("accountId");
        GetAccounts accs = new GetAccounts();
        String accountDetails = accs.getAccountDetail(accountId, context);
        String transactionDetails = accs.getTransactions(accountId, context);
        if(accountDetails != null) {
            model.addAttribute("accountDetails", accountDetails);
        }
        if(transactionDetails != null){
            model.addAttribute("transactionDetails", transactionDetails);
```

（6）Create APIContext.java, define inner class.



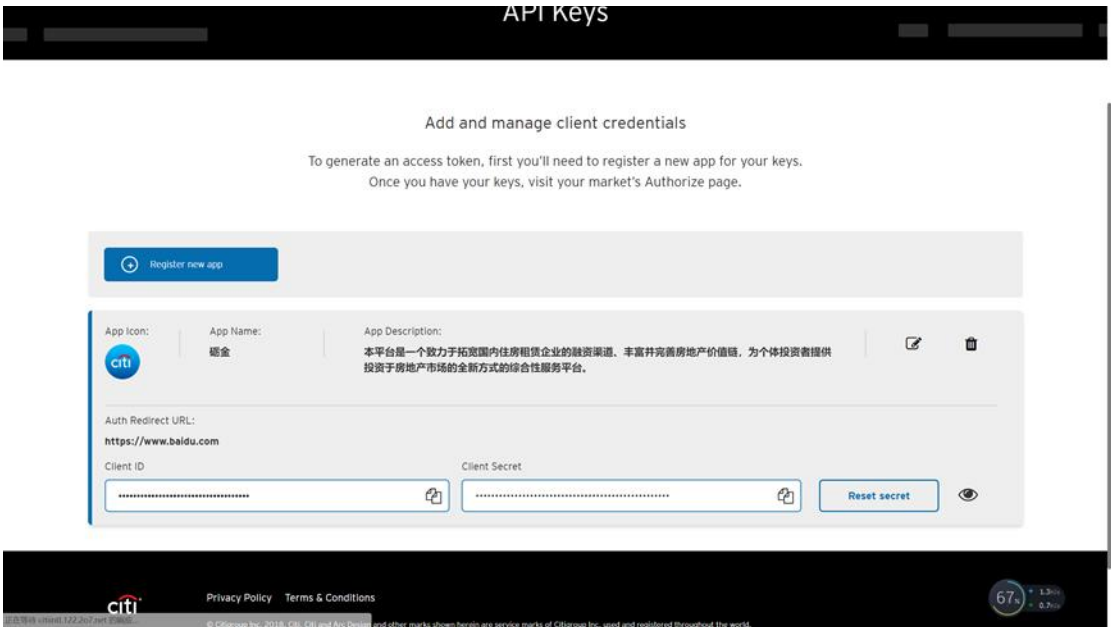
```java
package com.demo.developerapi.beans;

public class APIContext {
    private String accessToken;
    private String eventId;
    private String bizToken;
    private String realAccessToken;
    private String accounts;
    private String username;
    private String password;
    private String accountId;

    public String getAccessToken() {
        return accessToken;
    }

    public void setAccessToken(String accessToken) {
        this.accessToken = accessToken;
    }

    public String getEventId() {
        return eventId;
    }

    public void setEventId(String eventId) {
        this.eventId = eventId;
    }

    public String getBizToken() {
        return bizToken;
    }

    public void setBizToken(String bizToken) {
        this.bizToken = bizToken;
    }

    public String getRealAccessToken() {
        return realAccessToken;
    }

    public void setRealAccessToken(String realAccessToken) {
        this.realAccessToken = realAccessToken;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
```

（7）Create APIConstant.java, place client ID and client key.

```
package com.demo.developerapi.beans;

public interface APIConstant {
    final String CLIENT_ID = "4796196b-b36a-4067-89f5-bfde2e19812f";
    final String CLIENT_SCRENT = "1H1gN6aO1fF0mM5jW0pU2kV0hS6hU5bM4cU4iO3hR8iW7wS4hG";
}
```

3. Part Three: Functional Introduction

（1）Get Client Id/Client Secret and create app.


API Keys

[redacted]

Add and manage client credentials

To generate an access token, first you'll need to register a new app for your keys.
Once you have your keys, visit your market's Authorize page.

App Icon:

App Name: 砺金

App Description: 本平台是一个致力于拓宽国内住房租赁企业的融资渠道、丰富并完善房地产价值链，为个体投资者提供投资于房地产市场的全新方式的综合性服务平台。

Auth Redirect URL: https://www.baidu.com

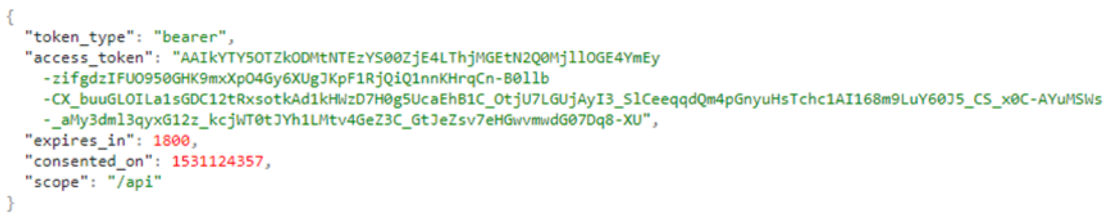
Client ID: ••••••••••••••••••••••••••••••••••••

Client Secret: ••

（2）Get client Access Token (actual running function).

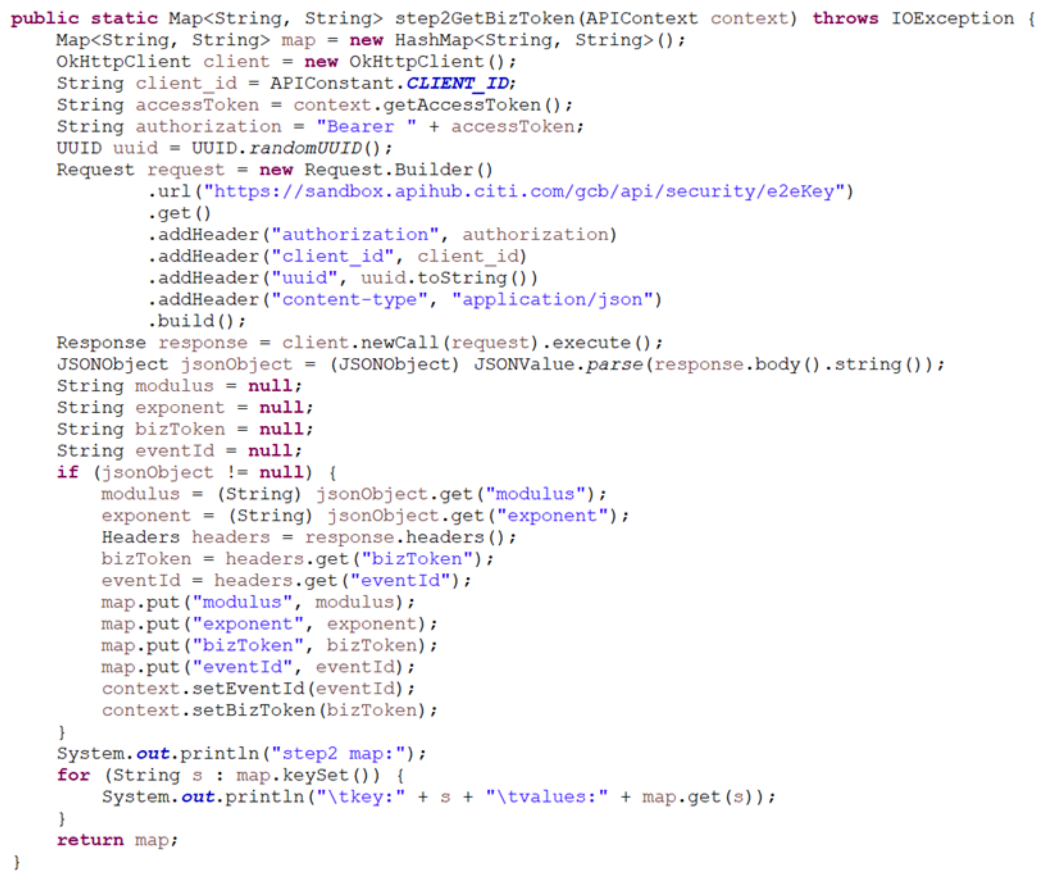
```
    public static String step1GetAccessToken(APIContext context) throws IOException {
        OkHttpClient client = new OkHttpClient();
        String client_id = APIConstant.CLIENT_ID;
        String client_scrent = APIConstant.CLIENT_SCRENT;
        String encode_key = client_id + ":" + client_scrent;
        String authorization = "Basic " + Base64.encodeBase64String(encode_key.getBytes());
        MediaType mediaType = MediaType.parse("application/x-www-form-urlencoded");
        RequestBody body = RequestBody.create(mediaType, "grant_type=client_credentials&scope=/api");
        Request request = new Request.Builder()
                .url("https://sandbox.apihub.citi.com/gcb/api/clientCredentials/oauth2/token/hk/gcb")
                .post(body)
                .addHeader("accept", "application/json")
                .addHeader("authorization", authorization)
                .addHeader("content-type", "application/x-www-form-urlencoded")
                .build();
        Response response = client.newCall(request).execute();
        JSONObject jsonObject = (JSONObject) JSONValue.parse(response.body().string());
        String accessToken = (String) jsonObject.get("access_token");
        context.setAccessToken(accessToken);
        System.out.println("step1 access_token:");
        System.out.println("\t" + accessToken);
        return accessToken;
    }
```

Return to the result：



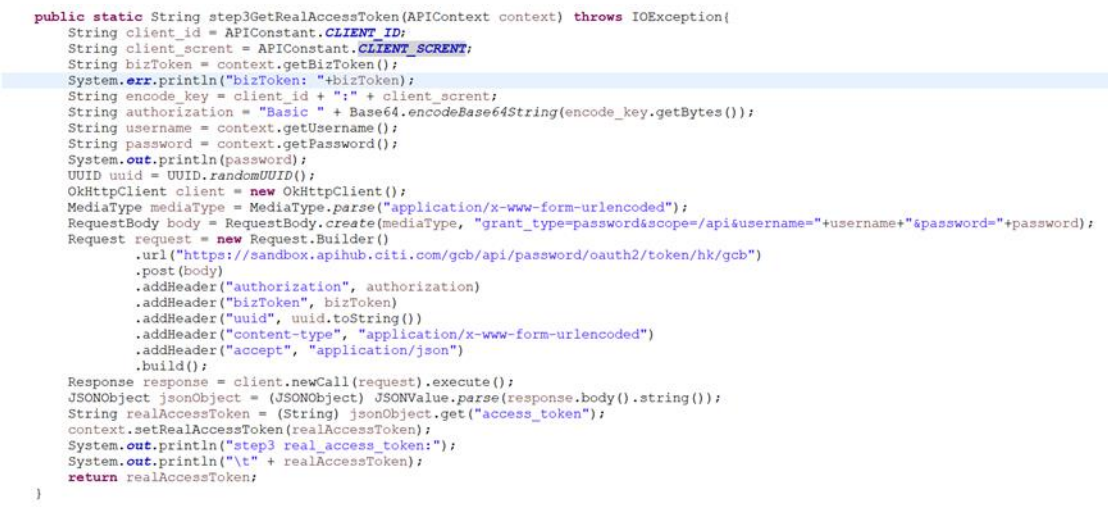
```json
{
  "token_type": "bearer",
  "access_token": "AAIkYTY5OTZkODMtNTEzYS00ZjE4LThjMGEtN2Q0MjllOGE4YmEy
    -zifgdzIFUO950GHK9mxXpO4Gy6XUgJKpF1RjQiQ1nnKHrqCn-B0llb
    -CX_buuGLOILa1sGDC12tRxsotkAd1kHWzD7H0g5UcaEhB1C_OtjU7LGUjAyI3_SlCeeqqdQm4pGnyuHsTchc1AI168m9LuY60J5_CS_x0C-AYuMSWs
    -_aMy3dml3qyxG12z_kcjWT0tJYh1LMtv4GeZ3C_GtJeZsv7eHGwvmwdG07Dq8-XU",
  "expires_in": 1800,
  "consented_on": 1531124357,
  "scope": "/api"
}
```

（3）Get modulus/exponent/bizToken/eventId (not actually running).



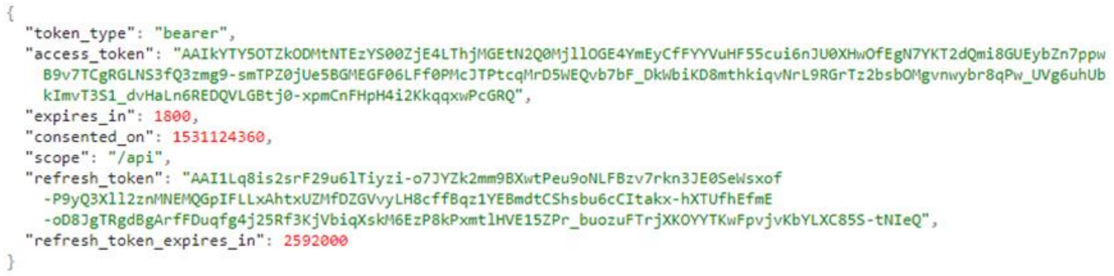
```java
public static Map<String, String> step2GetBizToken(APIContext context) throws IOException {
    Map<String, String> map = new HashMap<String, String>();
    OkHttpClient client = new OkHttpClient();
    String client_id = APIConstant.CLIENT_ID;
    String accessToken = context.getAccessToken();
    String authorization = "Bearer " + accessToken;
    UUID uuid = UUID.randomUUID();
    Request request = new Request.Builder()
            .url("https://sandbox.apihub.citi.com/gcb/api/security/e2eKey")
            .get()
            .addHeader("authorization", authorization)
            .addHeader("client_id", client_id)
            .addHeader("uuid", uuid.toString())
            .addHeader("content-type", "application/json")
            .build();
    Response response = client.newCall(request).execute();
    JSONObject jsonObject = (JSONObject) JSONValue.parse(response.body().string());
    String modulus = null;
    String exponent = null;
    String bizToken = null;
    String eventId = null;
    if (jsonObject != null) {
        modulus = (String) jsonObject.get("modulus");
        exponent = (String) jsonObject.get("exponent");
        Headers headers = response.headers();
        bizToken = headers.get("bizToken");
        eventId = headers.get("eventId");
        map.put("modulus", modulus);
        map.put("exponent", exponent);
        map.put("bizToken", bizToken);
        map.put("eventId", eventId);
        context.setEventId(eventId);
        context.setBizToken(bizToken);
    }
    System.out.println("step2 map:");
    for (String s : map.keySet()) {
        System.out.println("\tkey:" + s + "\tvalues:" + map.get(s));
    }
    return map;
}
```

（4）Access to Access Token (actual operation function).



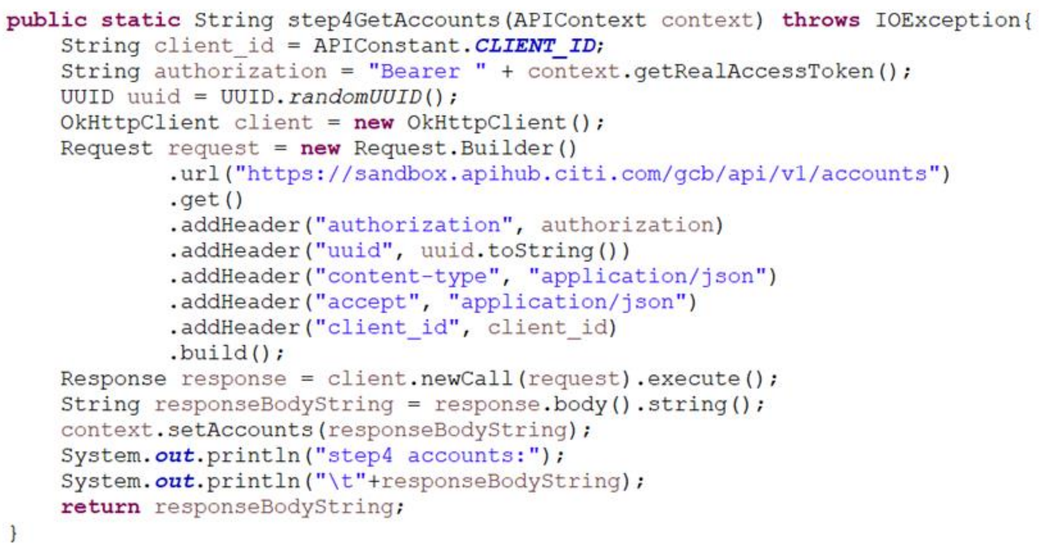
```java
public static String step3GetRealAccessToken(APIContext context) throws IOException{
    String client_id = APIConstant.CLIENT_ID;
    String client_scrent = APIConstant.CLIENT_SCRENT;
    String bizToken = context.getBizToken();
    System.err.println("bizToken: "+bizToken);
    String encode_key = client_id + ":" + client_scrent;
    String authorization = "Basic " + Base64.encodeBase64String(encode_key.getBytes());
    String username = context.getUsername();
    String password = context.getPassword();
    System.out.println(password);
    UUID uuid = UUID.randomUUID();
    OkHttpClient client = new OkHttpClient();
    MediaType mediaType = MediaType.parse("application/x-www-form-urlencoded");
    RequestBody body = RequestBody.create(mediaType, "grant_type=password&scope=/api&username="+username+"&password="+password);
    Request request = new Request.Builder()
            .url("https://sandbox.apihub.citi.com/gcb/api/password/oauth2/token/hk/gcb")
            .post(body)
            .addHeader("authorization", authorization)
            .addHeader("bizToken", bizToken)
            .addHeader("uuid", uuid.toString())
            .addHeader("content-type", "application/x-www-form-urlencoded")
            .addHeader("accept", "application/json")
            .build();
    Response response = client.newCall(request).execute();
    JSONObject jsonObject = (JSONObject) JSONValue.parse(response.body().string());
    String realAccessToken = (String) jsonObject.get("access_token");
    context.setRealAccessToken(realAccessToken);
    System.out.println("step3 real_access_token:");
    System.out.println("\t" + realAccessToken);
    return realAccessToken;
}
```

Return to the result：



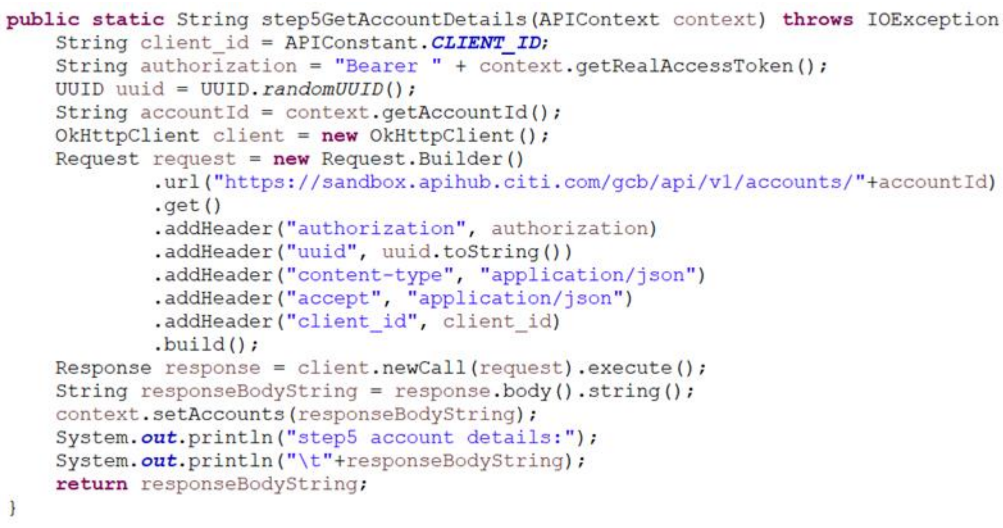
```
{
  "token_type": "bearer",
  "access_token": "AAIkYTY5OTZkODMtNTEzYS00ZjE4LThjMGEtN2Q0MjllOGE4YmEyCfFYYVuHF55cui6nJU0XHwOfEgN7YKT2dQmi8GUEybZn7ppw
    B9v7TCgRGLNS3fQ3zmg9-smTPZ0jUe5BGMEGF06LFf0PMcJTPtcqMrD5WEQvb7bF_DkWbiKD8mthkiqvNrL9RGrTz2bsbOMgvnwybr8qPw_UVg6uhUb
    kImvT3S1_dvHaLn6REDQVLGBtj0-xpmCnFHpH4i2KkqqxwPcGRQ",
  "expires_in": 1800,
  "consented_on": 1531124360,
  "scope": "/api",
  "refresh_token": "AAI1Lq8is2srF29u6lTiyzi-o7JYZk2mm9BXwtPeu9oNLFBzv7rkn3JE0SeWsxof
    -P9yQ3Xll2znMNEMQGpIFLLxAhtxUZMfDZGVvyLH8cffBqz1YEBmdtCShsbu6cCItakx-hXTUfhEfmE
    -oD8JgTRgdBgArfFDuqfg4j25Rf3KjVbiqXskM6EzP8kPxmtlHVE15ZPr_buozuFTrjXKOYYTKwFpvjvKbYLXC85S-tNIeQ",
  "refresh_token_expires_in": 2592000
}
```

（5）Get account information (not actually running).



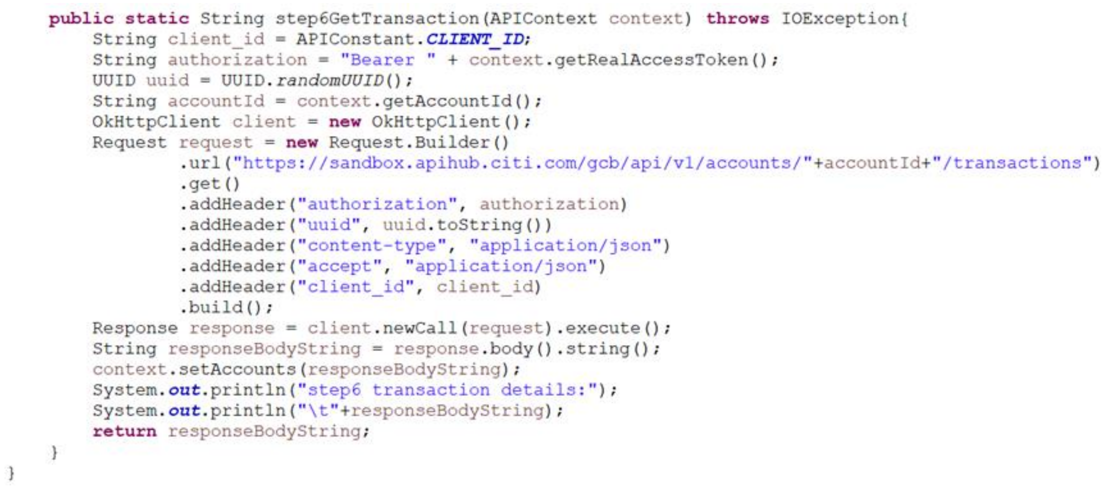
```
public static String step4GetAccounts(APIContext context) throws IOException{
    String client_id = APIConstant.CLIENT_ID;
    String authorization = "Bearer " + context.getRealAccessToken();
    UUID uuid = UUID.randomUUID();
    OkHttpClient client = new OkHttpClient();
    Request request = new Request.Builder()
            .url("https://sandbox.apihub.citi.com/gcb/api/v1/accounts")
            .get()
            .addHeader("authorization", authorization)
            .addHeader("uuid", uuid.toString())
            .addHeader("content-type", "application/json")
            .addHeader("accept", "application/json")
            .addHeader("client_id", client_id)
            .build();
    Response response = client.newCall(request).execute();
    String responseBodyString = response.body().string();
    context.setAccounts(responseBodyString);
    System.out.println("step4 accounts:");
    System.out.println("\t"+responseBodyString);
    return responseBodyString;
}
```

（6）Get the details of the account (not actually running).



```
public static String step5GetAccountDetails(APIContext context) throws IOException{
    String client_id = APIConstant.CLIENT_ID;
    String authorization = "Bearer " + context.getRealAccessToken();
    UUID uuid = UUID.randomUUID();
    String accountId = context.getAccountId();
    OkHttpClient client = new OkHttpClient();
    Request request = new Request.Builder()
            .url("https://sandbox.apihub.citi.com/gcb/api/v1/accounts/"+accountId)
            .get()
            .addHeader("authorization", authorization)
            .addHeader("uuid", uuid.toString())
            .addHeader("content-type", "application/json")
            .addHeader("accept", "application/json")
            .addHeader("client_id", client_id)
            .build();
    Response response = client.newCall(request).execute();
    String responseBodyString = response.body().string();
    context.setAccounts(responseBodyString);
    System.out.println("step5 account details:");
    System.out.println("\t"+responseBodyString);
    return responseBodyString;
}
```

（7）Get transaction information (not actually running).

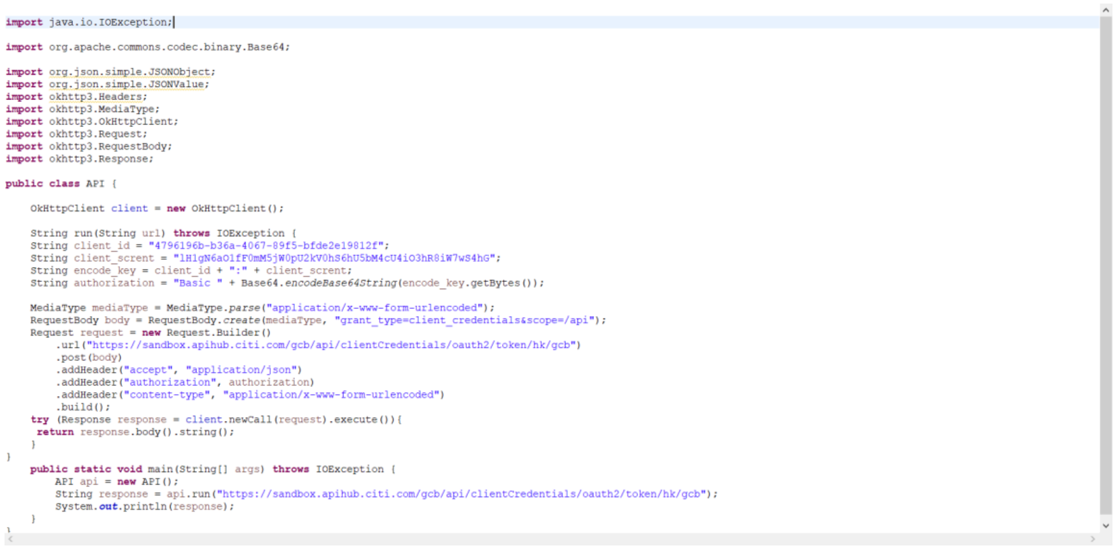


```
    public static String step6GetTransaction(APIContext context) throws IOException{
        String client_id = APIConstant.CLIENT_ID;
        String authorization = "Bearer " + context.getRealAccessToken();
        UUID uuid = UUID.randomUUID();
        String accountId = context.getAccountId();
        OkHttpClient client = new OkHttpClient();
        Request request = new Request.Builder()
                .url("https://sandbox.apihub.citi.com/gcb/api/v1/accounts/"+accountId+"/transactions")
                .get()
                .addHeader("authorization", authorization)
                .addHeader("uuid", uuid.toString())
                .addHeader("content-type", "application/json")
                .addHeader("accept", "application/json")
                .addHeader("client_id", client_id)
                .build();
        Response response = client.newCall(request).execute();
        String responseBodyString = response.body().string();
        context.setAccounts(responseBodyString);
        System.out.println("step6 transaction details:");
        System.out.println("\t"+responseBodyString);
        return responseBodyString;
    }
}
```

4.Part Four： Test Documentation

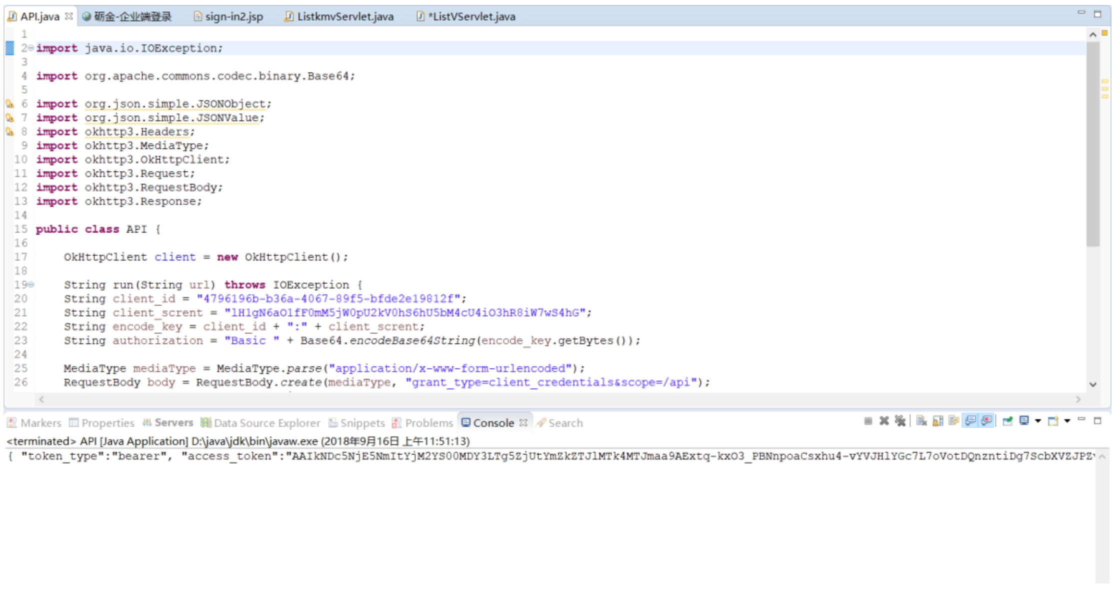
Platform A has written most of the functional code based on the introduction and code examples of the Citi Developer Portal. As shown in the third part above, all the functions are encapsulated into classes, provided that they are invoked and run in the background. Due to the weak correlation between the platform business and API and the urgency of time, this project only selects one function (to achieve access_token authorization for Citigroup users) to verify the test, and successfully returns the results.

The test code is as follows:



```
import java.io.IOException;

import org.apache.commons.codec.binary.Base64;

import org.json.simple.JSONObject;
import org.json.simple.JSONValue;
import okhttp3.Headers;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class API {

    OkHttpClient client = new OkHttpClient();

    String run(String url) throws IOException {
    String client_id = "4796196b-b36a-4067-89f5-bfde2e19812f";
    String client_scrent = "lH1gN6aO1fF0mM5jW0pU2kV0hS6hU5bM4cU4iO3hR8iW7wS4hG";
    String encode_key = client_id + ":" + client_scrent;
    String authorization = "Basic " + Base64.encodeBase64String(encode_key.getBytes());

    MediaType mediaType = MediaType.parse("application/x-www-form-urlencoded");
    RequestBody body = RequestBody.create(mediaType, "grant_type=client_credentials&scope=/api");
    Request request = new Request.Builder()
        .url("https://sandbox.apihub.citi.com/gcb/api/clientCredentials/oauth2/token/hk/gcb")
        .post(body)
        .addHeader("accept", "application/json")
        .addHeader("authorization", authorization)
        .addHeader("content-type", "application/x-www-form-urlencoded")
        .build();
    try (Response response = client.newCall(request).execute()){
     return response.body().string();
    }
}
    public static void main(String[] args) throws IOException {
        API api = new API();
        String response = api.run("https://sandbox.apihub.citi.com/gcb/api/clientCredentials/oauth2/token/hk/gcb");
        System.out.println(response);
    }
}
```

The results are as follows:



```java

import java.io.IOException;

import org.apache.commons.codec.binary.Base64;

import org.json.simple.JSONObject;
import org.json.simple.JSONValue;
import okhttp3.Headers;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class API {

    OkHttpClient client = new OkHttpClient();

    String run(String url) throws IOException {
    String client_id = "4796196b-b36a-4067-89f5-bfde2e19812f";
    String client_scrent = "lH1gN6aO1fF0mM5jW0pU2kV0hS6hU5bM4cU4iO3hR8iW7wS4hG";
    String encode_key = client_id + ":" + client_scrent;
    String authorization = "Basic " + Base64.encodeBase64String(encode_key.getBytes());

    MediaType mediaType = MediaType.parse("application/x-www-form-urlencoded");
    RequestBody body = RequestBody.create(mediaType, "grant_type=client_credentials&scope=/api");
```

```
<terminated> API [Java Application] D:\java\jdk\bin\javaw.exe (2018年9月16日 上午11:51:13)
{ "token_type":"bearer", "access_token":"AAIkNDc5NjE5NmItYjM2YS00MDY3LTg5ZjUtYmZkZTJlMTk4MTJmaa9AExtq-kxO3_PBNnpoaCsxhu4-vYVJHlYGc7L7oVotDQnzntiDg7ScbXVZJPZ
```

As you can see, the system has successfully returned access_token; at this point, the Platform successfully implemented the call function to the Citigroup API.