

# Pinelabs Java SDK Integration

This document explains the integration process of the Pinelabs Java SDK for your Java Applications

By Pinelabs Team

## Prerequisites

Open JDK Version >=17

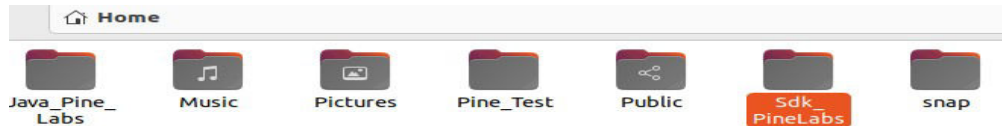
## Installation Process

This section explains the installation flow of the Java SDK for dealing with Pine Lab Apis

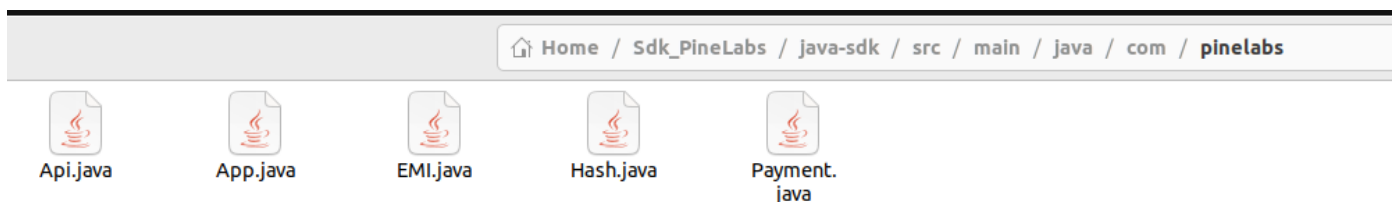
In this example we've created a sample project named `Java\_Pine\_Labs` which will be provided to you along with the SDK for you to be able to run and test out the SDK without actually integrating it. You can change and modify the code to check different behavior and responses we get from the SDK.

Given below are the steps for installing the SDK in your project:

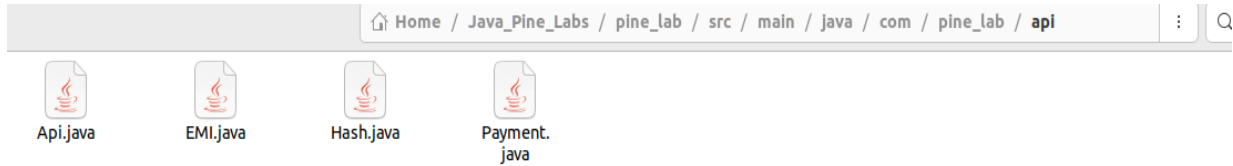
1. Extract the SDK (Zip File) in any location on your system. Here we have extracted it inside Sdk\_PineLabs Folder (Sdk Folder - java-sdk)



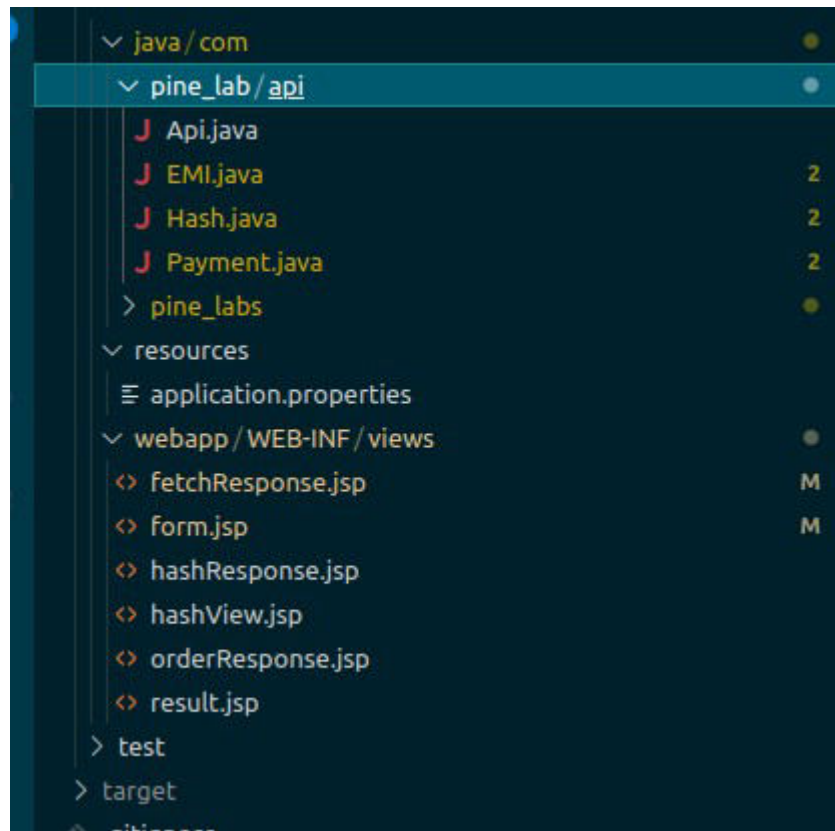
2. Now we need to come into this path `src/main/java/com/pine_lab/api` inside `java-sdk` folder and in that we'll be able to locate all the necessary classes, which will be imported in your current project directory. So basically you can copy these 4 files (`Api.java`, `EMI.java`, `Hash.java`, `Payment.java`) and create new package in your java framework and add these files or you can add into your existing package if class name is not repeated.



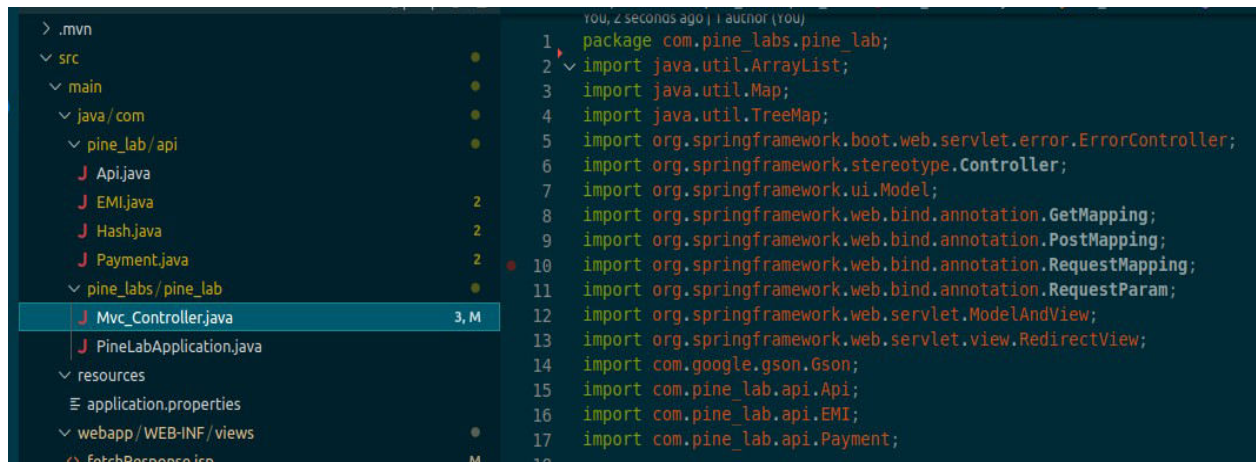
- For sample we have imported files in our Sample Springboot Project (Java\_Pine\_Labs), and created a new package pine\_lab/api and injected all classes of sdk into it.



- So it'll look something like this in your IDE, where your main classes are also located in other packages. In this other main controller classes are located in com/pine\_labs & sdk files have been added into com/pine\_lab/api.



5. After that you need to import classes into your main controller class, like in our Sample Springboot Project we have included classes (import com.pine\_lab.api.Api;) in that way you need to import other classes as well. If and only if you have added sdk into another package else you don't need to add import statement into it.



```
you, 2 seconds ago | 1 author (you)
1 package com.pine_lab.pine_lab;
2 import java.util.ArrayList;
3 import java.util.Map;
4 import java.util.TreeMap;
5 import org.springframework.boot.web.servlet.error.ErrorController;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.ui.Model;
8 import org.springframework.web.bind.annotation.GetMapping;
9 import org.springframework.web.bind.annotation.PostMapping;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.RequestParam;
12 import org.springframework.web.servlet.ModelAndView;
13 import org.springframework.web.servlet.view.RedirectView;
14 import com.google.gson.Gson;
15 import com.pine_lab.api.Api;
16 import com.pine_lab.api.EMI;
17 import com.pine_lab.api.Payment;
```

6. Then you need to copy all the dependencies in your current project's Pom.xml file from Sdk's Pom.xml file. For reference below image shows xml file of SDK.



```
<dependency>
  <groupId>com.squareup.okhttp3</groupId>
  <artifactId>okhttp</artifactId>
  <version>4.9.1</version>
</dependency>
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.13.0</version>
</dependency>
<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.3.1</version>
</dependency>
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpclient</artifactId>
  <version>4.5.13</version>
</dependency>
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.8.8</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>4.0.1</version>
  <scope>provided</scope>
</dependency>
```

## Usage Process

This section explains the usage flow of the Java SDK in Spring-Boot Framework.

**(Note-** For testing this Spring-Boot sample app, you need to run main class and hit the url on browser to test all 4 methods <http://localhost:8081/form>, Port number mentioned 8081 has been declared in Application Properties, so you can change according to requirements)

Now we have the classes of SDK imported into our Project's newly created Package named api in path com/pine\_lab

1. Api Class- Used for initializing object containing Merchant details so we don't have to repeat process of passing these details into every class.
2. Payment Class – Used for creating payment links and fetching payment details for a transaction id
3. EMI Class – Used for calculation of EMI data for a product
4. Hash Class – Used for hash verification in your application for secure development

## Create Order Process

In order for us to create an order and generate a URL for payment we need to use the “Payment Class”,

1. Create an instance of the Api class first and pass the following parameters
  - a. Merchant Id of the merchant provided by Pinelabs team (Integer)
  - b. Merchant Access Code also provided by Pinelabs team (String)
  - c. Merchant Secret also provided by Pinelabs team (String)
  - d. Pg Mode which will be of type boolean (True for Testing and False for Production)

```
Api apip = new Api(merchant_id, access_code, secret, pg_mode);
```

2. Now you need to call payment method with Api class's object and assign it to Payment class's instance and then you need to call the “Create” method on the instance of the Payment class and pass the following arguments
  - a. Txn Id- Required, String
  - b. Amount- Long
  - c. Callback Url- String
  - d. Customer Data – Required: Customer information including the first name, last name, mobile number, customer id, email, Shipping Data, Billing Data etc. (Map Data Structure)
  - e. Billing Data – Optional: Billing details of the customer (Map Data Structure)
  - f. UDF Data – Optional: UDF data which can be anything extra you want to include in the order (Map Data Structure)
  - g. Payment Modes – Required: Modes of payments you want to be enabled on the checkout using the payment (ArrayList of Strings)

**Below is the image showing how to pass data, so basically this is response of html form through which we're fetching details and passing it to our SDK's functions according to required format.**

```

Boolean pg_mode = Boolean.parseBoolean((String) map.get("pg_mode"));
String secret = (String) map.get("secret");
String access_code = (String) map.get("access_code");
Integer merchant_id = Integer.parseInt((String) map.get("merchant_id"));
ArrayList<String> paymentModes = new ArrayList<String>();
TreeMap<String, Object> customerData = new TreeMap<>();
TreeMap<String, String> billing_data = new TreeMap<>();

```

```

String customer_id = (String) map.get("customer_id");
customer_id.trim();
if (customer_id.length() != 0) {
    customerData.put("customer_id", customer_id);
}
String email_id = (String) map.get("email");
email_id.trim();
if (email_id.length() != 0) {
    customerData.put("email_id", email_id);
}
String first_name = (String) map.get("first_name");
first_name.trim();
if (first_name.length() != 0) {
    customerData.put("first_name", first_name);
}
String last_name = (String) map.get("last_name");
last_name.trim();
if (last_name.length() != 0) {
    customerData.put("last_name", last_name);
}
String mobile_no = (String) map.get("phone");
mobile_no.trim();
if (mobile_no.length() != 0) {
    customerData.put("mobile_no", mobile_no);
}
String address1 = (String) map.get("address1");
address1.trim();
if (address1.length() != 0) {
    billing_data.put("address1", address1);
}
String address2 = (String) map.get("address2");
address2.trim();
if (address2.length() != 0) {
    billing_data.put("address2", address2);
}
String address3 = (String) map.get("address3");
address3.trim();
if (address3.length() != 0) {
    billing_data.put("address3", address3);
}

```

```

customerData.put("billing_data", billing_data);
customerData.put("shipping_data", shipping_data);

```

```

Api apip = new Api(merchant_id, access_code, secret, pg_mode);
Payment obj = apip.payment();
Map<String, Object> paymentResponse = obj.create(txnId, amount, callback_url, customerData, udfData,
    paymentModes);

```



3. It will return the following once executed
  - a. **Token:** Payment order token for processing of order
  - b. **URL:** URL using which the user can be directly redirected to the payment page where he/she can make the payment.
  - c. **Status:** Status of the order create api status if it worked successfully or not.

```
{
  "token": "S01%2fGkV5TCwLcFCrLw4qeJ%2f8aLDtoQhzGS%2b%2fiNar3AXEbQ%3d",
  "url": "https://uat.pinepg.in/pinepg/v2/process/payment?token=S01%2fGkV5TCwLcFCrLw4qeJ%2f8aLDtoQhzGS%2b%2fiNar3AXEbQ%3d",
  "status": true
}
```

## Fetch Order Process

In order for us to fetch an order detail we need to use the “Payment Class”

1. We’ll call the “Fetch” method and pass it the following
  - a. Transaction ID: Transaction id which was used during the order creation process explained at the top (string)
  - b. Transaction Type: Type of the transaction which was passed at the order creation process. This needs to be the same otherwise it won’t work. (Integer)

Below is the image showing how to pass data, so basically this is response of html form through which we’re fetching details and passing it to our SDK’s functions according to required format.

```
Api apip = new Api(Integer.parseInt(merchant_id), access_code, secret, Boolean.parseBoolean(mode));
Payment obj = apip.payment();
Map<String, Object> enquiryResponse = obj.fetch(txn_id, txnType:3);
```

2. Resopnse

```
{
  "ppc_MerchantID": "106600",
  "ppc_MerchantAccessCode": "bcf441be-411b-46a1-aa88-c6e852a7d68c",
  "ppc_PinePGTxnStatus": "7",
  "ppc_TransactionCompletionDateTime": "20/09/2023 04:07:52 PM",
  "ppc_UniqueMerchantTxnID": "650acb67d3752",
  "ppc_Amount": "1000",
  "ppc_TxnResponseCode": "1",
  "ppc_TxnResponseMessage": "SUCCESS",
  "ppc_PinePGTransactionID": "12069839",
  "ppc_CapturedAmount": "1000",
  "ppc_RefundedAmount": "0",
  "ppc_AcquirerName": "BILLDESK",
  "ppc_DIA_SECRET": "A36BFCDE7745E7D889C6B57FD8FC4F5861DC74805B4C42A7414A9BE3A928543D",
  "ppc_DIA_SECRET_TYPE": "SHA256",
  "ppc_PaymentMode": "3",
  "ppc_Parent_TxnStatus": "4",
  "ppc_ParentTxnResponseCode": "1",
  "ppc_ParentTxnResponseMessage": "SUCCESS",
  "ppc_CustomerMobile": "7737291210",
  "ppc_UdfField1": "",
  "ppc_UdfField2": "",
  "ppc_UdfField3": "",
  "ppc_UdfField4": "",
  "ppc_AcquirerResponseCode": "0300",
  "ppc_AcquirerResponseMessage": "DEFAULT"
}
```

## EMI Details Process

In order for us to fetch EMI details for any product we need to use the “EMI Class”

1. First create an instance of the class and refer it to Api class’s object calling emiCalculation method, like shown below and pass it the following parameters
  - a. Merchant Id of the merchant provided by Pinelabs team
  - b. Merchant Access Code also provided by Pinelabs team
  - c. Merchant Secret also provided by Pinelabs team
  - d. Pg mode (True for Testing and False for Production)

2. You need to pass product\_details as list of map (For Multi or Single Cart), and in map it should have product\_code, product\_amount as keys as shown in image below, and along with that you need to pass total amount as passed in amount variable.

```
ArrayList<TreeMap<String, Object>> product_details = new ArrayList<TreeMap<String, Object>>();
TreeMap<String, Object> products = new TreeMap<String, Object>();
products.put("product_code", product_code1);
products.put("product_amount", Long.parseLong(amount1));
product_details.add(products);
if(product_code2 != "" && amount2 != ""){
TreeMap<String, Object> products2 = new TreeMap<String, Object>();
products2.put("product_code", product_code2);
products2.put("product_amount", Long.parseLong(amount2));
product_details.add(products2);
}
```

```
Api apip = new Api(Integer.parseInt(merchant_id), access_code, secret, Boolean.parseBoolean(mode));
EMI obj = apip.emi();
Map<String, Object> emiResponse = obj.emiCalculation(Long.parseLong(amount), product_details);
```

3. Once executed it will return the following response



## Hash Verification Process

In order for us to verify hash of all the responses we get from the SDK to make sure there was no tampering with the response we need to use the “Payment Class”

1. First create an instance of the class and refer it to Api class’s object calling vrifyHash method, like shown below and pass it the following parameters
  - a. Merchant Id of the merchant provided by Pinelabs team
  - b. Merchant Access Code also provided by Pinelabs team
  - c. Merchant Secret also provided by Pinelabs team
  - d. Pg mode (True for Testing and False for Production)
2. Pass the following params to the method
  - a. **Response Body:** Pass the entire response body **without the hash and hash type**



```

Api apip = new Api(merchant_id, access_code, secret, pg_mode);
Payment obj = apip.payment();
TreeMap<String, Object> dataMap = new TreeMap<>();
System.out.println((String) map.get("ppc_MerchantID"));
dataMap.put("ppc_MerchantID", (String) map.get("merchant_id"));
dataMap.put("ppc_MerchantAccessCode", (String) map.get("access_code"));
dataMap.put("ppc_PinePGTxnStatus", (String) map.get("ppc_PinePGTxnStatus"));
dataMap.put("ppc_TransactionCompletionDateTime", (String) map.get("ppc_TransactionCompletionDateTime"));
dataMap.put("ppc_UniqueMerchantTxnID", (String) map.get("txn_id"));
dataMap.put("ppc_Amount", (String) map.get("ppc_Amount"));
dataMap.put("ppc_TxnResponseCode", (String) map.get("ppc_TxnResponseCode"));
dataMap.put("ppc_TxnResponseMessage", (String) map.get("ppc_TxnResponseMessage"));
dataMap.put("ppc_PinePGTransactionID", (String) map.get("ppc_PinePGTransactionID"));
dataMap.put("ppc_CapturedAmount", (String) map.get("ppc_CapturedAmount"));
dataMap.put("ppc_RefundedAmount", (String) map.get("ppc_RefundedAmount"));
dataMap.put("ppc_AcquirerName", (String) map.get("ppc_AcquirerName"));
dataMap.put("ppc_PaymentMode", (String) map.get("ppc_PaymentMode"));
dataMap.put("ppc_Parent_TxnStatus", (String) map.get("ppc_Parent_TxnStatus"));
dataMap.put("ppc_ParentTxnResponseCode", (String) map.get("ppc_ParentTxnResponseCode"));
dataMap.put("ppc_ParentTxnResponseMessage", (String) map.get("ppc_ParentTxnResponseMessage"));
dataMap.put("ppc_CustomerMobile", (String) map.get("ppc_CustomerMobile"));
if (map.get("ppc_UdfField1") == null) {
    dataMap.put("ppc_UdfField1", "");
} else {
    dataMap.put("ppc_UdfField1", map.get("ppc_UdfField1"));
}
if (map.get("ppc_UdfField2") == null) {
    dataMap.put("ppc_UdfField2", "");
} else {
    dataMap.put("ppc_UdfField2", map.get("ppc_UdfField2"));
}
if (map.get("ppc_UdfField3") == null) {
    dataMap.put("ppc_UdfField3", "");
} else {
    dataMap.put("ppc_UdfField3", map.get("ppc_UdfField3"));
}
if (map.get("ppc_UdfField4") == null) {
    dataMap.put("ppc_UdfField4", "");
} else {
    dataMap.put("ppc_UdfField4", map.get("ppc_UdfField4"));
}
dataMap.put("ppc_AcquirerResponseCode", (String) map.get("ppc_AcquirerResponseCode"));
dataMap.put("ppc_AcquirerResponseMessage", (String) map.get("ppc_AcquirerResponseMessage"));
boolean hashVerification = obj.verifyHash((String) map.get("hash"), dataMap);

```

### TDR/GST Information:

Please note no additional charges like TDR, GST, etc are handled in our Plugins and the same needs to be manually handled at the merchant end.

### TLS 1.2 information:

Java 8 and above provide TLS 1.2 support.

Note: The exact version might depend on the specific Java vendor (Oracle, OpenJDK, etc.).