# **Thanh toán tích hợp Paypal**

Tham khảo bài viết gốc trên code project :

* <https://www.codeproject.com/Articles/870870/Using-Paypal-Rest-API-with-ASP-NET-MVC>
* <https://code.tutsplus.com/articles/paypal-integration-part-2-paypal-rest-api--cms-22917>

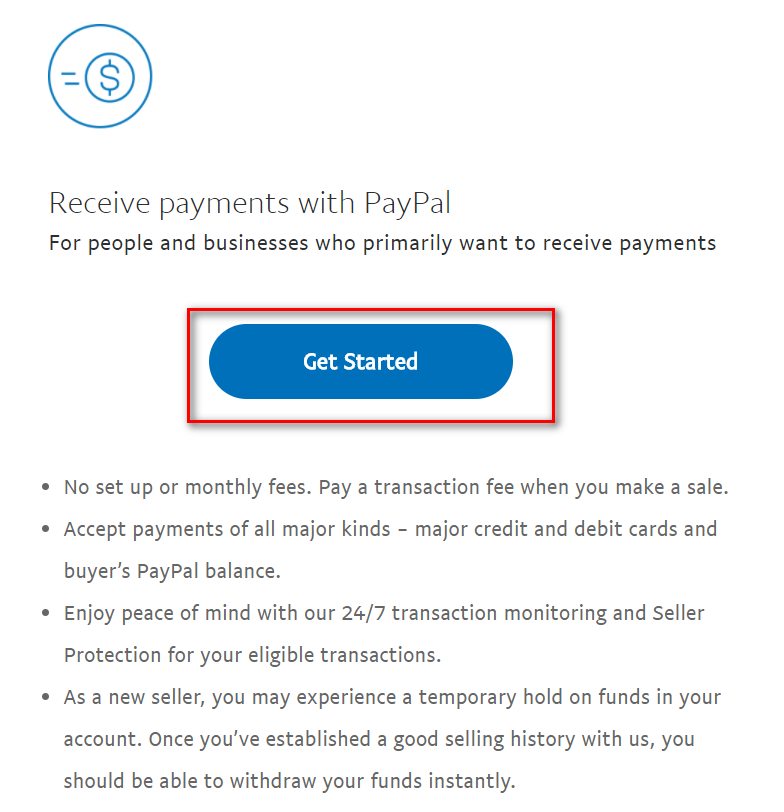
Bài viết có tham khảo và điều chỉnh bỏ bớt một số nội dung không cần thiết, cũng như hướng dẫn và giải thích chi tiết việc tạo tài khoản paypal và môi trường tích hợp để kiểm thử sandbox

**Nội dung bổ sung thêm:**

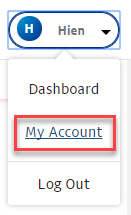
* Hướng dẫn đăng ký tài khoản tích hợp thử trên môi trường sandbox.
* Hướng dẫn chuyển từ môi trường test sandbox sang môi trường thanh toán thật trên paypal.
* Hướng dẫn cách lấy **ClientId** và **ClientSecret** key.

## Đăng ký tài khoản tích hợp cho nhà phát triển

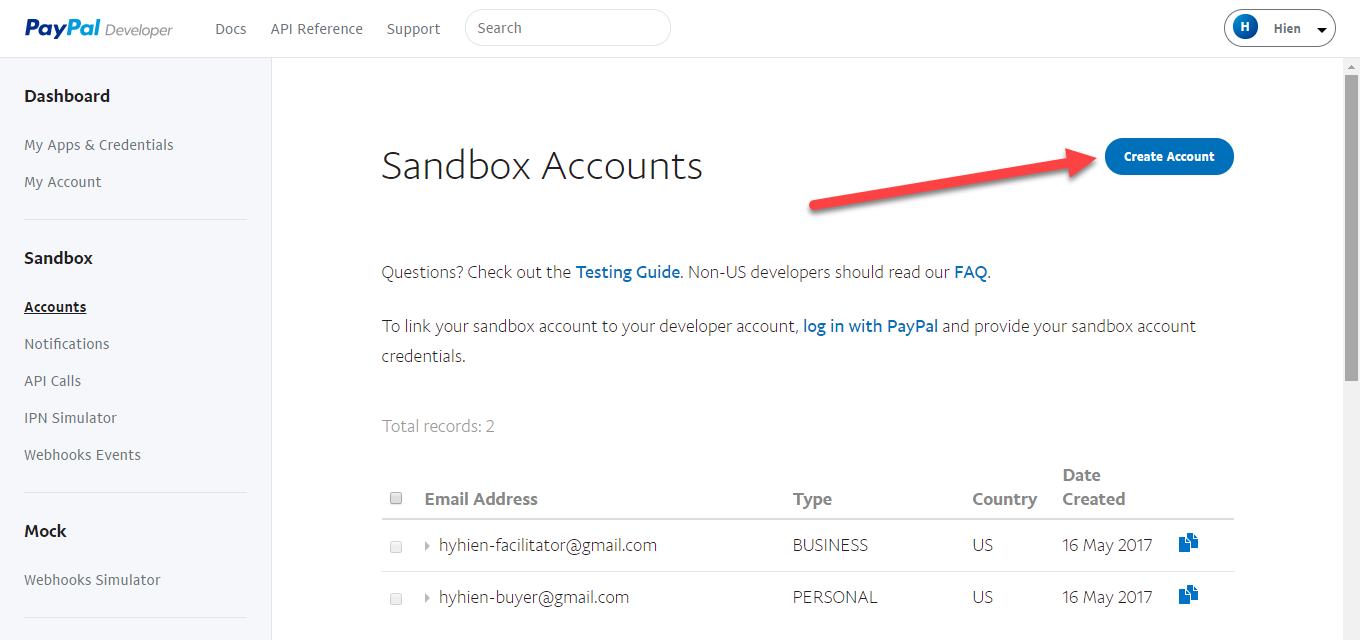
* <https://developer.paypal.com>

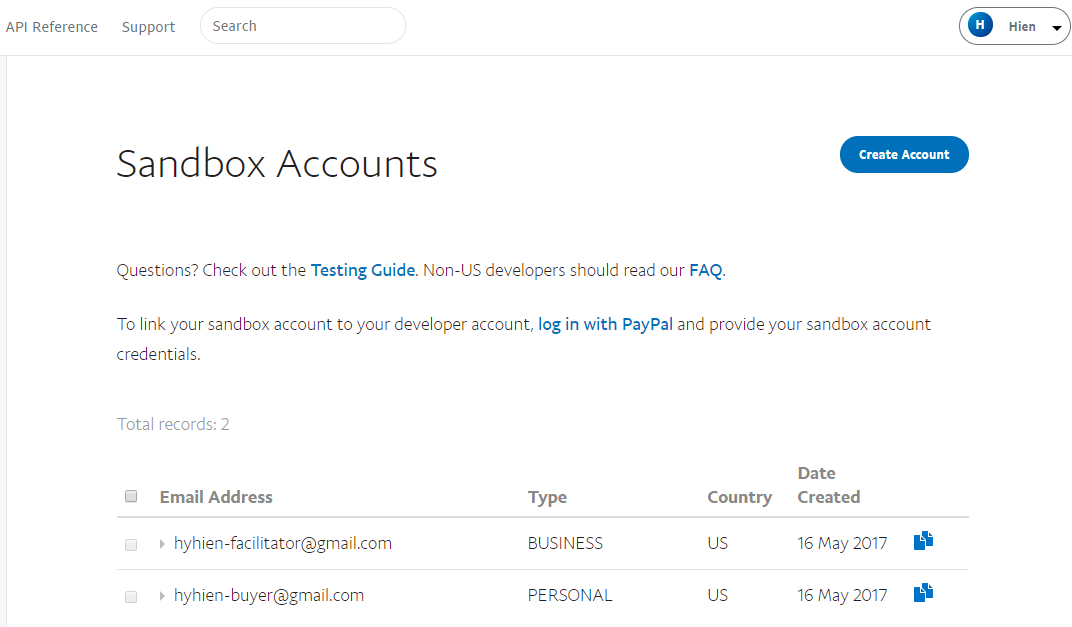


* Xác nhận email
* Đăng ký tích hợp ứng dụng vào website

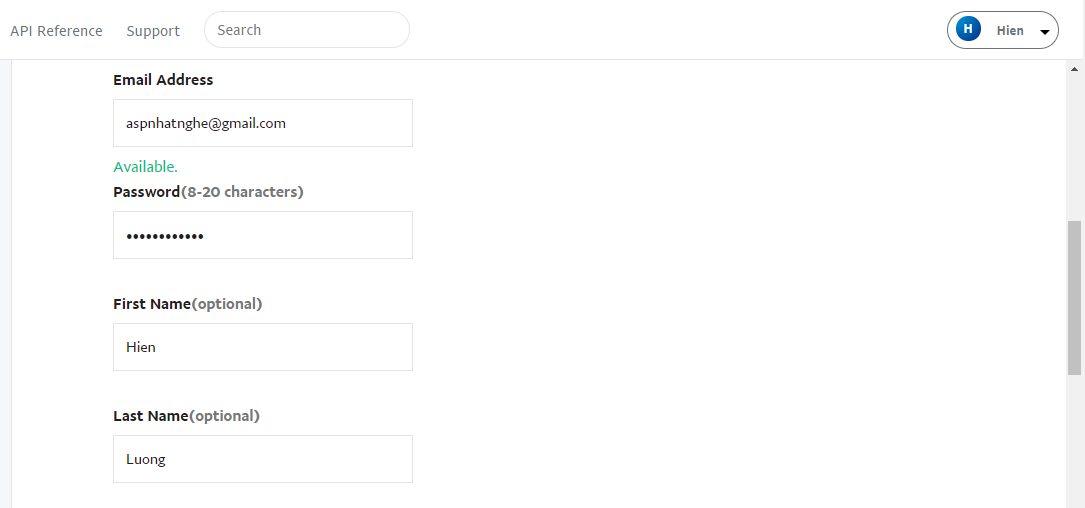


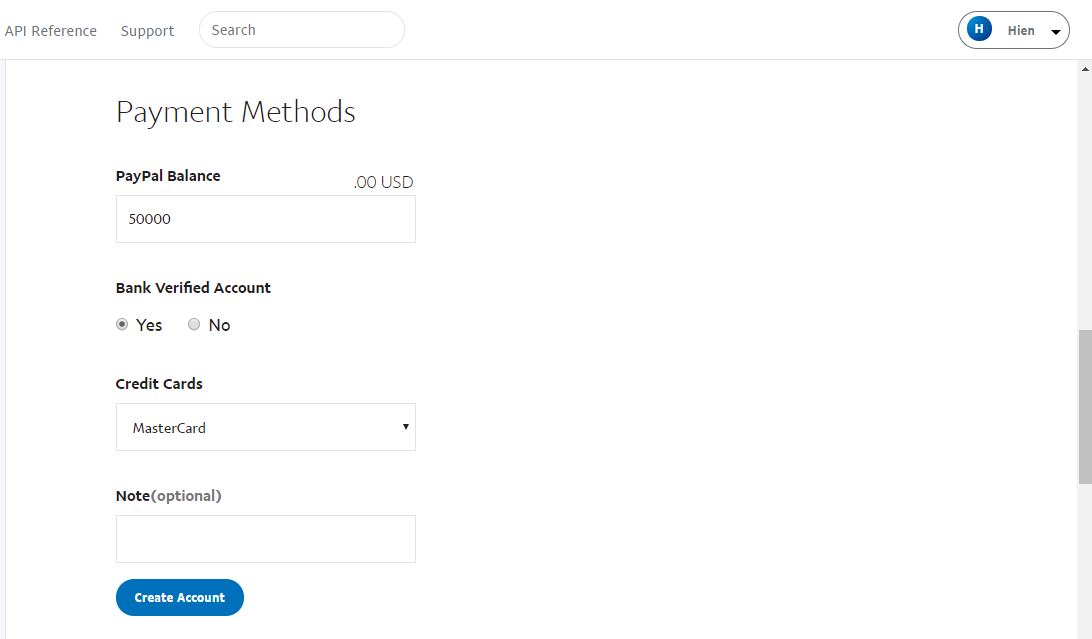
* Tạo thêm account Sandbox dành cho người bán:



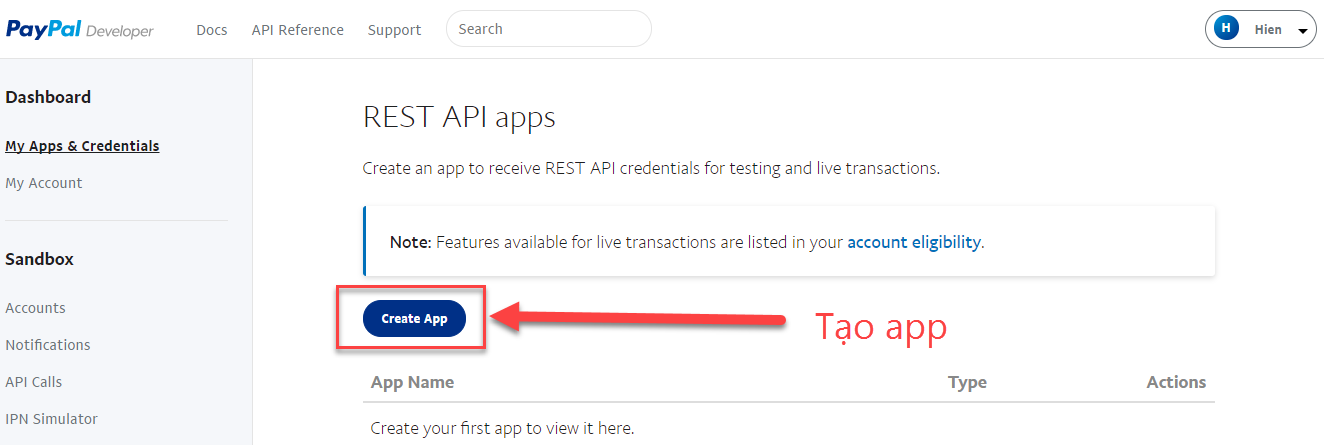


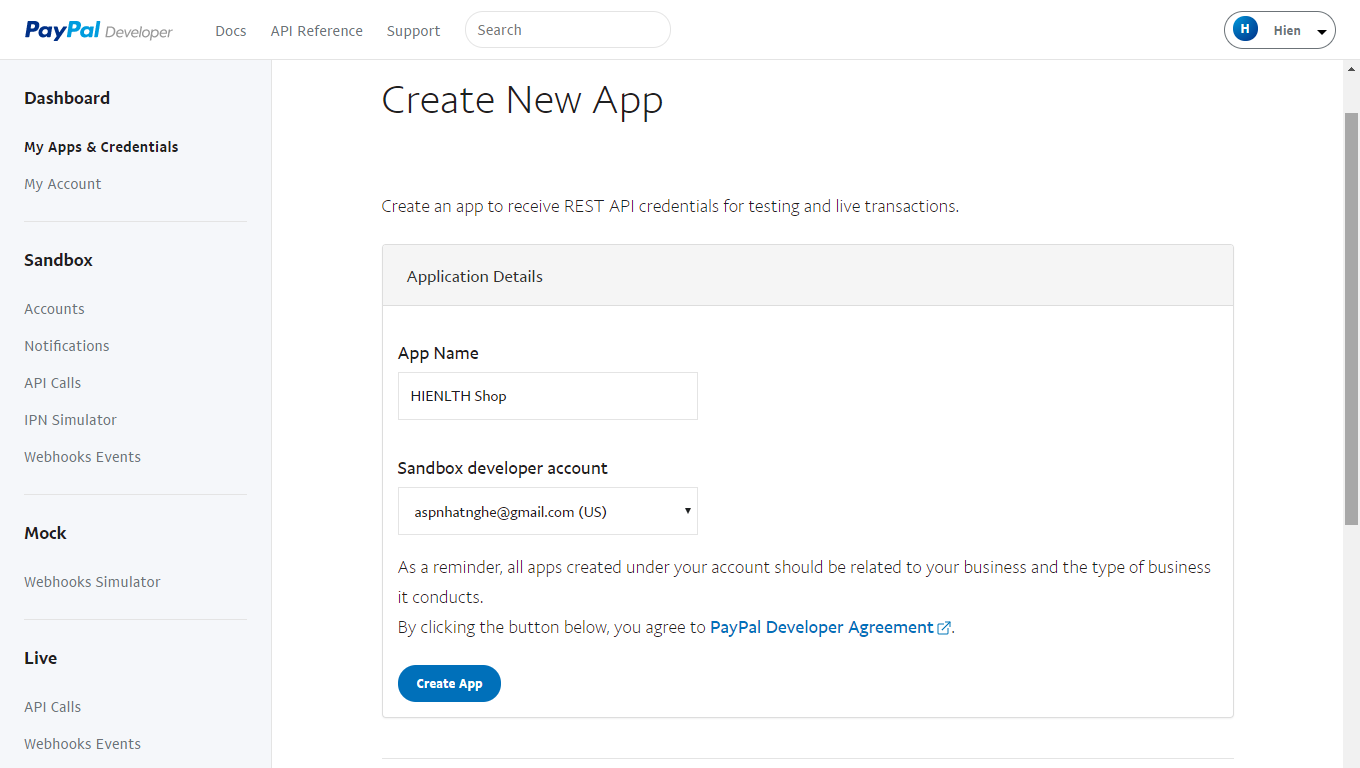
Các bạn có quyền tạo tài khoản khác.





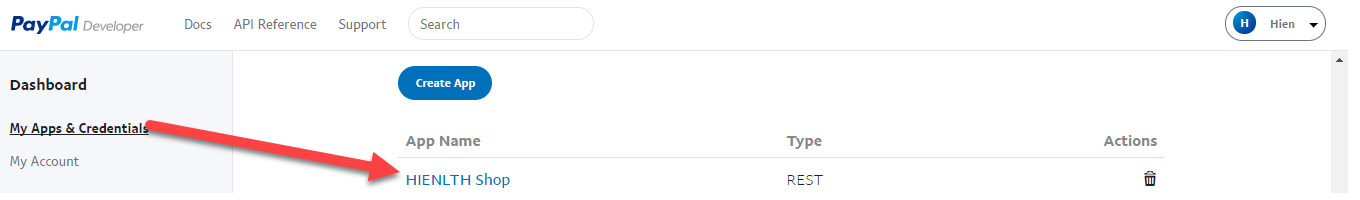
* Tạo App:



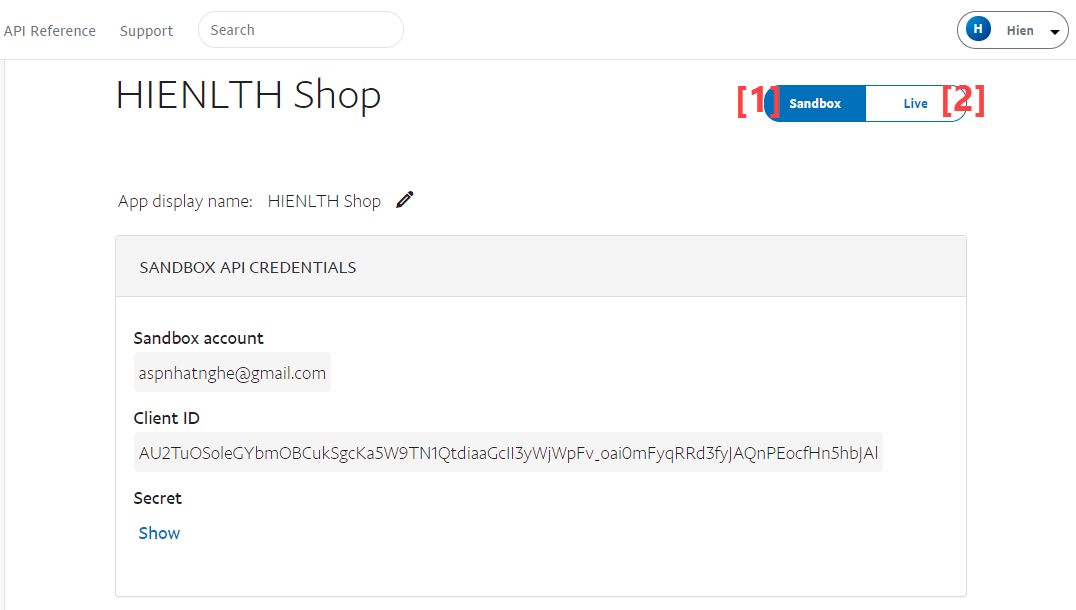


* Lấy thông tin Client ID và Client Secret key.

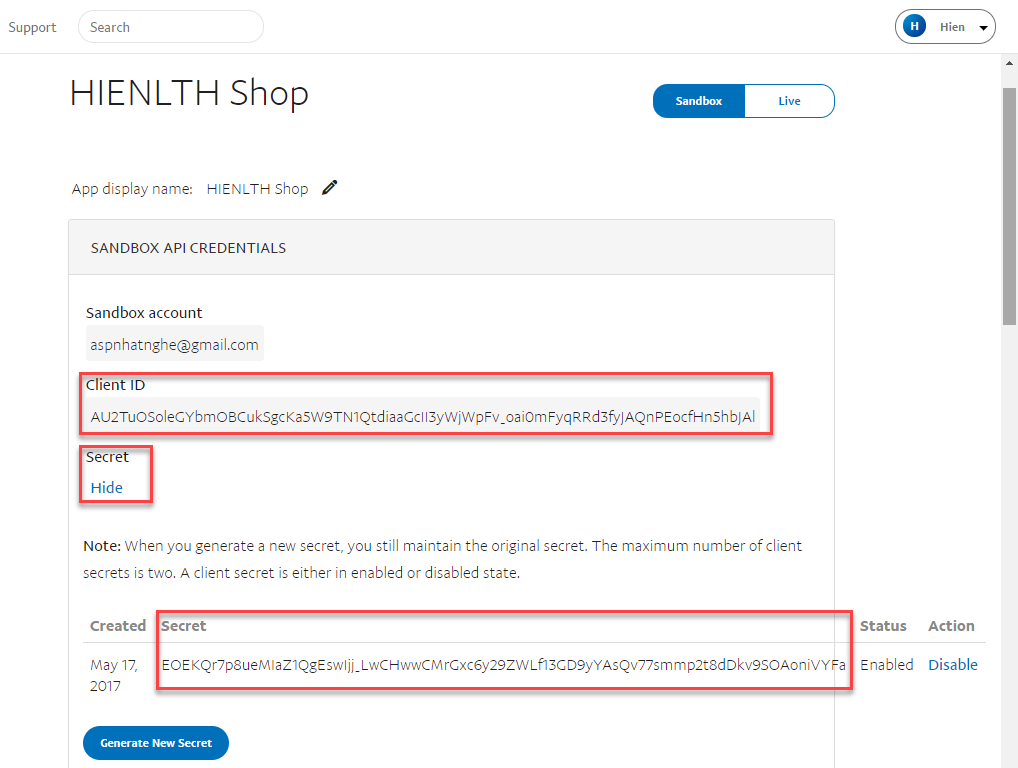
**Lưu ý: Tài khoản *sandbox*:** paypal tạo môi trường thanh toán ảo tại **sandbox.paypal.com** cho nhà phát triển gần giống với môi trường thanh toán và quản lý tiền thật **paypal.com,** nhà phát triển cần phải kiểm tra kỹ lưỡng trước khi đưa ra môi trường triển khai thật.



* Vào App



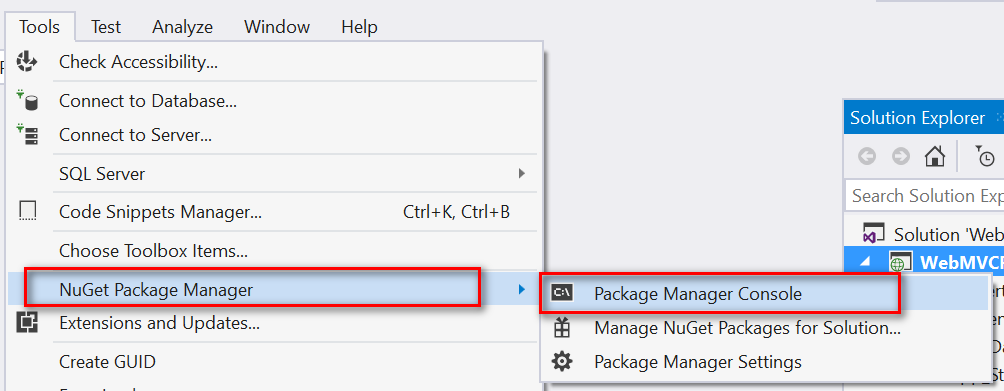
* [1] Môi trường kiểm thử tích hợp ảo cho nhà phát triển
* [2] Môi trường tích hợp thật



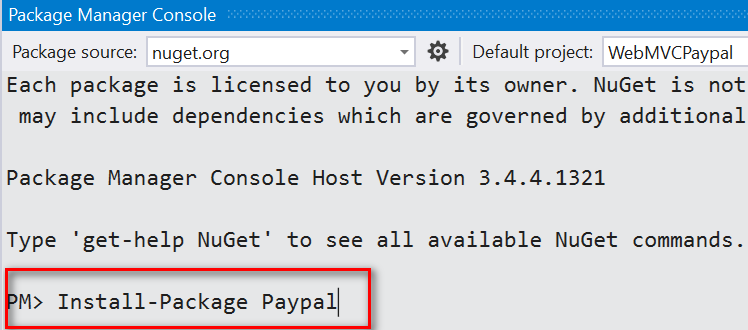
* Lưu **Client ID và Client Secret key** vào file text document (txt) để làm các bước tiếp theo

## Tích hợp vào ứng dụng website (lưu ý: bước này đã có sẵn website bán hàng cần tích hợp thanh toán paypal hoặc tự tạo ứng dụng website ASP.NET MVC để thực hiện)

* Tạo mới project ASP NET MVC Web Application 🡪 đặt tên ứng dụng.
* Thêm thư viện **Paypal SDK for .NET** thông qua **NuGet Package Manager** hoặc thêm từ viện PayPal.dll, Newtonsoft.Json.dll từ thư mục <<thu-vien>> kèm theo bài hướng dẫn
* Chọn Tools -> Library Package Manager -> Package Manager Console

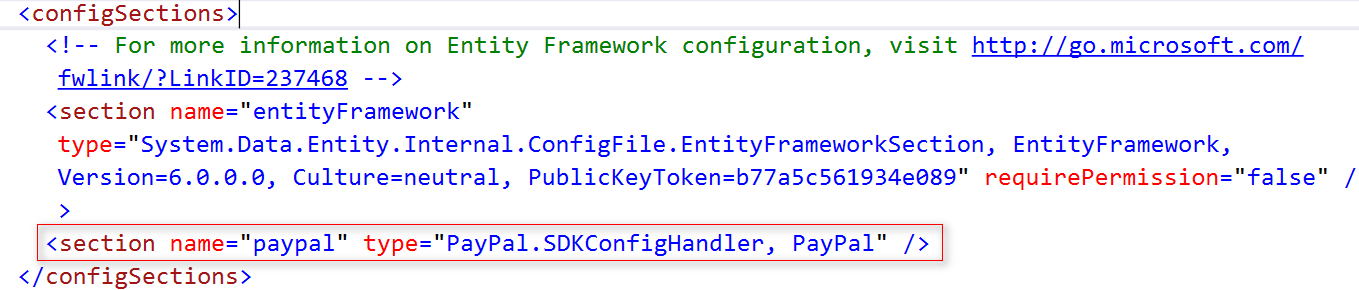


* + Install-Package Paypal



## Cấu hình file web.config bao gồm các thông tin thanh toán từ paypal

* **Client ID và Client Secret key** lấy ở bước trên (môi trường sandbox)



* Code

|  |
| --- |
| *<section name="paypal" type="PayPal.SDKConfigHandler, PayPal" />* |

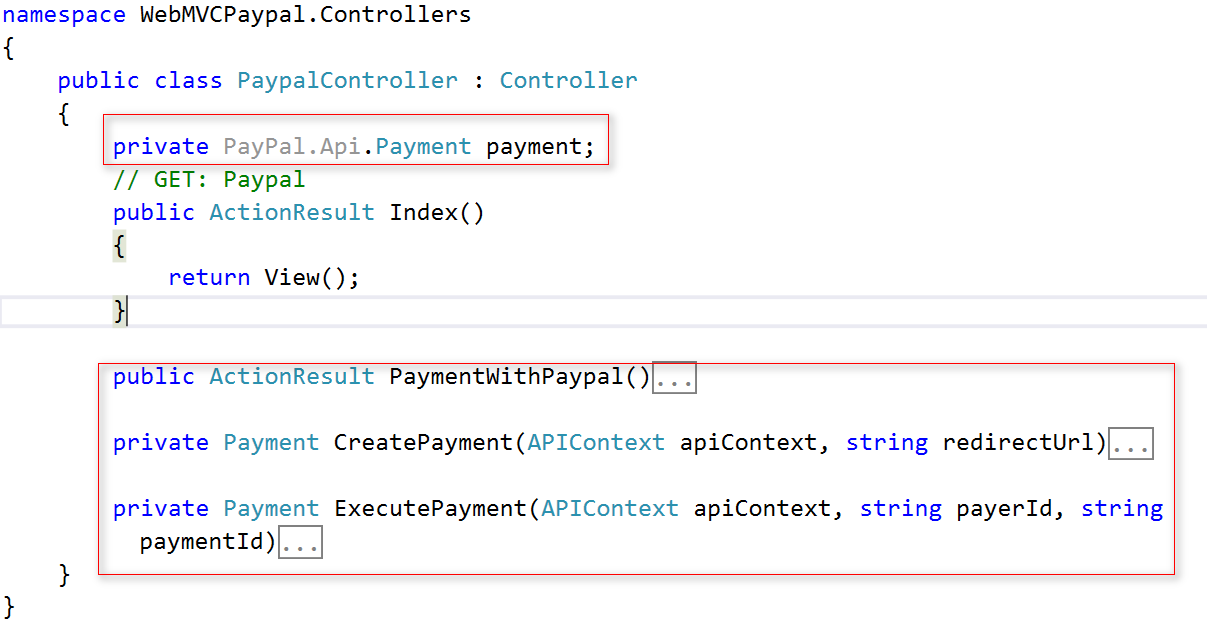
* Tiếp theo



* Code

|  |
| --- |
| *<paypal>*  *<settings>*  *<add name="mode" value="sandbox" />*  *<add name="connectionTimeout" value="360000" />*  *<add name="requestRetries" value="1" />*  *<add name="clientId" value="Gia tri clientId" />*  *<add name="clientSecret" value="Gia tri Secret" />*  *</settings>*  *</paypal>* |

## Tạo controller thực hiện thanh toán qua **Paypal** đặt tên **PaypalController.cs**



|  |
| --- |
| public ActionResult PaymentWithPaypal()  {  //getting the apiContext as earlier  APIContext apiContext = Configuration.GetAPIContext();  try  {  string payerId = Request.Params["PayerID"];  if (string.IsNullOrEmpty(payerId))  {  //this section will be executed first because PayerID doesn't exist  //it is returned by the create function call of the payment class  // Creating a payment  // baseURL is the url on which paypal sendsback the data.  // So we have provided URL of this controller only  string baseURI = Request.Url.Scheme + "://" + Request.Url.Authority +  "/Paypal/PaymentWithPayPal?";  //guid we are generating for storing the paymentID received in session  //after calling the create function and it is used in the payment execution  var guid = Convert.ToString((new Random()).Next(100000));  //CreatePayment function gives us the payment approval url  //on which payer is redirected for paypal account payment  var createdPayment = this.CreatePayment(apiContext, baseURI + "guid=" + guid);  //get links returned from paypal in response to Create function call  var links = createdPayment.links.GetEnumerator();  string paypalRedirectUrl = null;  while (links.MoveNext())  {  Links lnk = links.Current;  if (lnk.rel.ToLower().Trim().Equals("approval\_url"))  {  //saving the payapalredirect URL to which user will be redirected for payment  paypalRedirectUrl = lnk.href;  }  }  // saving the paymentID in the key guid  Session.Add(guid, createdPayment.id);  return Redirect(paypalRedirectUrl);  }  else  {  // This section is executed when we have received all the payments parameters  // from the previous call to the function Create  // Executing a payment  var guid = Request.Params["guid"];  var executedPayment = ExecutePayment(apiContext, payerId, Session[guid] as string);  if (executedPayment.state.ToLower() != "approved")  {  return View("FailureView");  }  }  }  catch (Exception ex)  {  Logger.Log("Error" + ex.Message);  return View("FailureView");  }  return View("SuccessView");  } |

|  |
| --- |
| private Payment CreatePayment(APIContext apiContext, string redirectUrl)  {  var itemList = new ItemList() { items = new List<Item>() };  //Các giá trị bao gồm danh sách sản phẩm, thông tin đơn hàng  //Sẽ được thay đổi bằng hành vi thao tác mua hàng trên website  itemList.items.Add(new Item()  {  //Thông tin đơn hàng  name = "Item Name",  currency = "USD",  price = "5",  quantity = "1",  sku = "sku"  });  //Hình thức thanh toán qua paypal  var payer = new Payer() { payment\_method = "paypal" };  // Configure Redirect Urls here with RedirectUrls object  var redirUrls = new RedirectUrls()  {  cancel\_url = redirectUrl,  return\_url = redirectUrl  };  //các thông tin trong đơn hàng  var details = new Details()  {  tax = "1",  shipping = "1",  subtotal = "5"  };  //Đơn vị tiền tệ và tổng đơn hàng cần thanh toán  var amount = new Amount()  {  currency = "USD",  total = "7", // Total must be equal to sum of shipping, tax and subtotal.  details = details  };  var transactionList = new List<Transaction>();  //Tất cả thông tin thanh toán cần đưa vào transaction  transactionList.Add(new Transaction()  {  description = "Transaction description.",  invoice\_number = "your invoice number",  amount = amount,  item\_list = itemList  });  this.payment = new Payment()  {  intent = "sale",  payer = payer,  transactions = transactionList,  redirect\_urls = redirUrls  };  // Create a payment using a APIContext  return this.payment.Create(apiContext);  } |

|  |
| --- |
| private Payment ExecutePayment(APIContext apiContext, string payerId, string paymentId)  {  var paymentExecution = new PaymentExecution() { payer\_id = payerId };  this.payment = new Payment() { id = paymentId };  return this.payment.Execute(apiContext, paymentExecution);  } |

## Tạo thư mục /Helper, thêm 2 file Configuration.cs và Logger.cs

* Configuration.cs

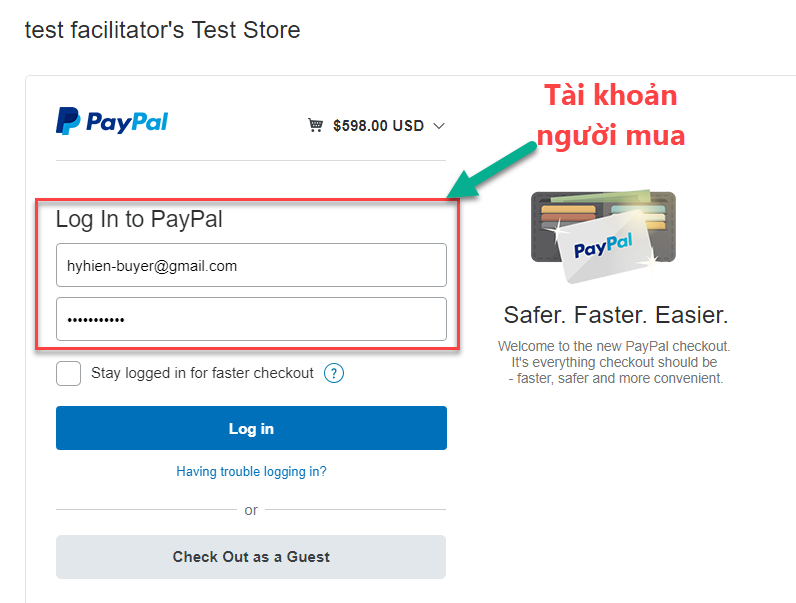
|  |
| --- |
| public static class Configuration  {  public readonly static string ClientId;  public readonly static string ClientSecret;  // Static constructor for setting the readonly static members.  static Configuration()  {  var config = GetConfig();  ClientId = config["clientId"];  ClientSecret = config["clientSecret"];  }  // Create the configuration map that contains mode and other optional configuration details.  public static Dictionary<string, string> GetConfig()  {  return PayPal.Api.ConfigManager.Instance.GetProperties();  }  // Create accessToken  private static string GetAccessToken()  {  // ###AccessToken  // Retrieve the access token from  // OAuthTokenCredential by passing in  // ClientID and ClientSecret  // It is not mandatory to generate Access Token on a per call basis.  // Typically the access token can be generated once and  // reused within the expiry window  string accessToken = new OAuthTokenCredential(ClientId, ClientSecret, GetConfig()).GetAccessToken();  return accessToken;  }  // Returns APIContext object  public static APIContext GetAPIContext()  {  // ### Api Context  // Pass in a `APIContext` object to authenticate  // the call and to send a unique request id  // (that ensures idempotency). The SDK generates  // a request id if you do not pass one explicitly.  APIContext apiContext = new APIContext(GetAccessToken());  apiContext.Config = GetConfig();  // Use this variant if you want to pass in a request id  // that is meaningful in your application, ideally  // a order id.  // String requestId = Long.toString(System.nanoTime();  // APIContext apiContext = new APIContext(GetAccessToken(), requestId ));  return apiContext;  }  } |

* Logger.cs

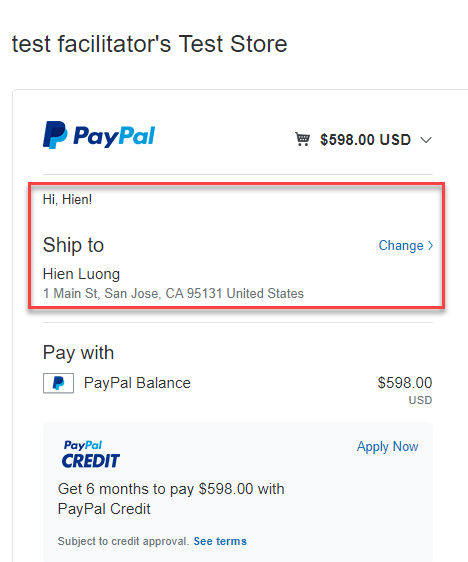
|  |
| --- |
| public class Logger  {  public static string LogDirectoryPath = Environment.CurrentDirectory;  public static void Log(String lines)  {  // Write the string to a file.append mode is enabled so that the log  // lines get appended to test.txt than wiping content and writing the log  try  {  System.IO.StreamWriter file = new System.IO.StreamWriter(LogDirectoryPath + "\\Error.log", true);  file.WriteLine(DateTime.Now.ToString("dd/MM/yyyy HH:mm:ss") + " --> " + lines);  file.Close();  }  catch  {    }  }  } |

## Chạy thử ứng dụng tích hợp

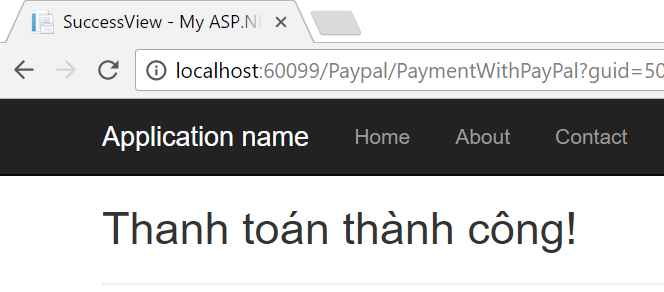
* Http:// localhost:60099/paypal/**PaymentWithPaypal**



* Xử lý

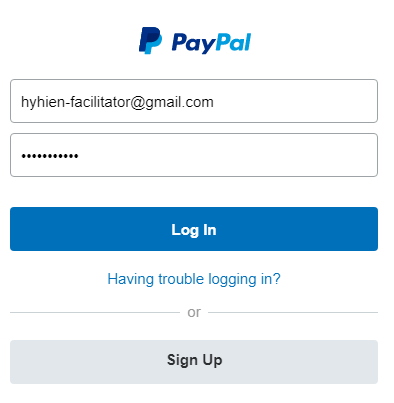


* Kết quả

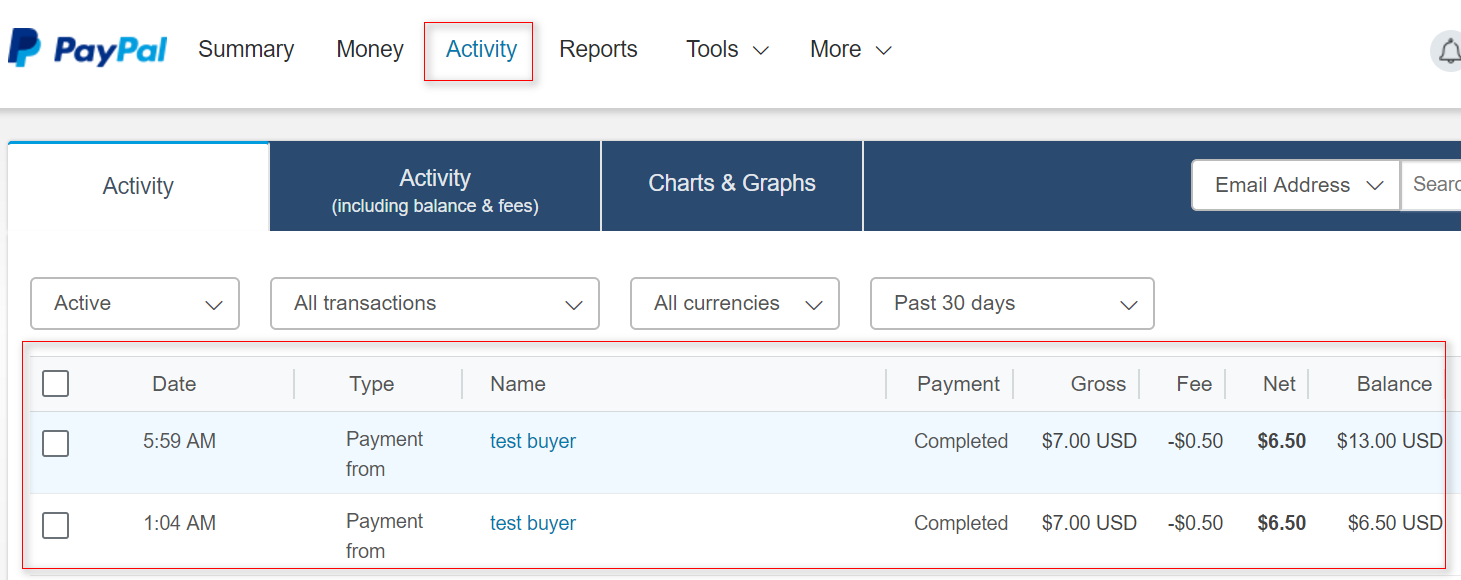


* Đăng nhập vào <https://www.sandbox.paypal.com/> để kiểm tra đơn hàng

Sử dụng tài khoản người bán + người mua để kiểm tra:



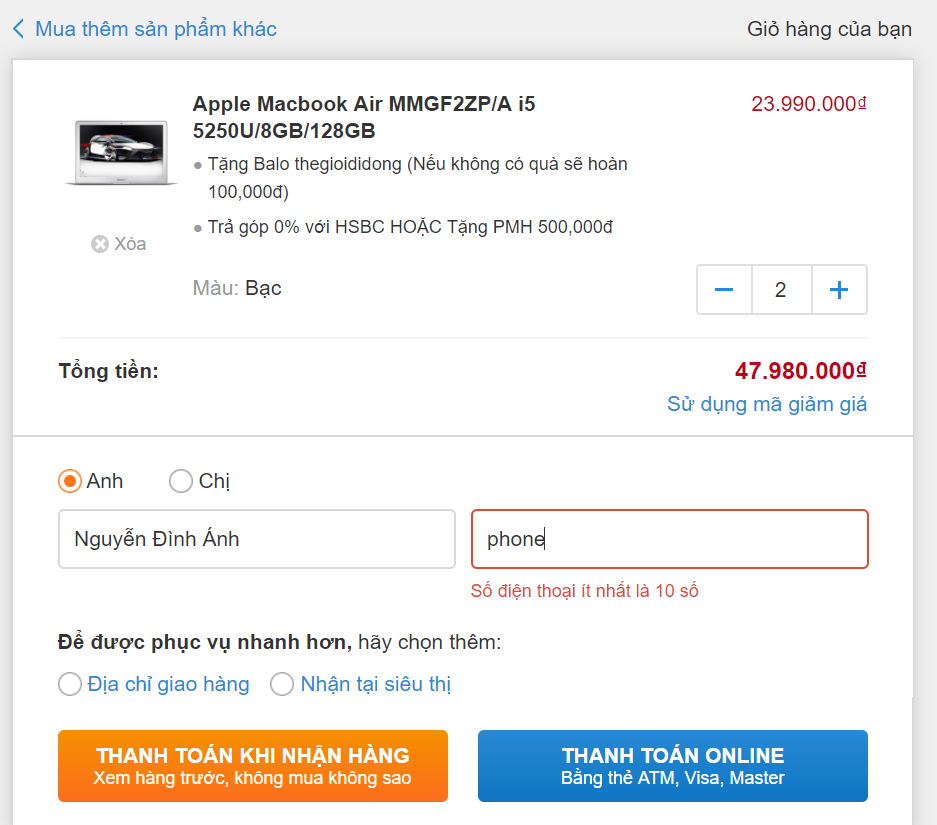
* Xem thống kê đơn hàng



* Sau khi Kiểm thử thành công trong môi trường Sandbox, chuyển đổi thành môi trường thật để thanh toán paypal (sinh viên tự thực hiện theo mô tả ở phần trên)

## Bổ sung: Sinh viên hiện thực các màn hình

* Thiết kế màn hình xem giỏ hàng, người dùng chọn hình thức thanh toán qua paypal



* Hệ thống nhận các giá trị của sản phẩm từ CSDL và thông tin khách hàng để thanh toán online qua paypal như hướng dẫn tại các phần trước.