

This Free E-Book is brought to you by Natural-Aging.com.

100% Effective Natural Hormone Treatment
Menopause, Andropause And Other Hormone Imbalances
Impair Healthy Healing In People Over The Age Of 30!

Designing and Building Payment Applications

By Robert Levings

Designing and Building Payment Applications by Robert Levings

DESIGNING AND BUILDING PAYMENT APPLICATIONS

By Robert Levings, President, EasyPay123

Unless you are using an off-the-shelf shopping cart application, chances are that you or your developer will be customizing at least some element of your payment application to meet your specific needs. Your payment application may be simple or it may be complex; either way, following the right sequence of events and taking advantage of available technology will help increase customer satisfaction; reduce development costs, payment-related errors and time to market; and provide you with a more functional application.

The following article highlights the key steps involved in designing, building and launching payment applications. While it is written from a developer's perspective, it will equip the merchant with information that you can use to when you are interacting with your developer throughout the process. It is part of a series of articles offered by EasyPay123 to help merchants understand the many facets of processing credit card payments.

Step 1: Understand User Needs

Understanding user needs is a necessary starting point for the development of any payment application. Documenting how the user will interact with the application and all of its various features ensures that all parties can agree on what the final product will look like and how it will operate. Some of the questions that must be answered at this stage include:

- * How will payments be accepted (online, phone-in, IVR, etc.)?
- * What customer data is to be captured (e.g. name, phone #, address info, etc.)?
- * What supplemental data will be captured (e.g. demographics, questionnaire, etc.)?
- * Which data fields should be made "required" on the interface?
- * What error detection/correction or data validation scripts will be used (e.g. MOD10 credit card number validation, date/dollar format, etc.)?

Designing and Building Payment Applications

- * Do any calculations need to be made on the interface?
- * What will the look and feel of the interface be (graphics, text, colors and layout)?
- * Will access to the application be restricted by IP address or username/password?
- * Will the results of the transactions update a database in real-time or be made accessible to a third party application/data-base at some point after the transactions take place?
- * How will the credit card data be input (keyed, swiped, touch screen, etc.)?
- * What content should be put on the response/receipt page?
- * How will the customer/merchant receive an email receipt (if desired)?
- * What will the contents of the emailed receipt be?
- * What bank(s) will be used for merchant accounts?
- * Where will the application be hosted?
- * How will the payment form be secured?

The answers to these questions should be captured in a specification document so that the merchant can review and comment. This step will reduce development time and cost, and also increase merchant satisfaction with the project.

Step 2: Design the Interface

Designing the user interface is always "part art/part technology" and will vary greatly depending upon the application. It should be driven primarily from the user needs defined above. In addition, technical considerations, such as the target operating system, hosting environment and development language must be taken into account. These decisions are the "meat and potatoes" of most developers' business, so little discussion is necessary here. An interface mock-up is useful so that users can provide input and test out the application (some payment gateways provide developer environments so that a transaction can be accurately simulated with the application prior to "going live").

You may want to include any number of scripts in the interface to reduce errors and improve its functionality. Some that you may find useful include:

- (a) MOD10 Check: Uses the Luhn algorithm to determine whether the credit card number is valid or not. Reduces transaction time. Could reduce merchant cost if gateway charges for failed credit card numbers.
- (b) Order Number Generator: Generates a unique numeric order number that can be posted with each transaction. With the "reject duplicate order number" option turned on (if supported), it nearly eliminates the risk of duplicate transactions being processed by the gateway.
- (c) Required Fields: Disables the ability to submit a transaction unless all required fields are completed on the order form. Ensures all necessary / desired data is posted to the payment gateway.
- (d) Double Submit Prevention: Disables the ability for the user to click the "Submit" button again while a transaction is still processing. Reduces the risk of an erroneous double submission of an order.
- (e) Email Validation: Ensures that an email address is in the format XXXXX@YYYY.ZZZ. Reduces the risk of incorrect emails being entered.
- (f) Complete If Left Empty: Automatically populated a specific data field if the user chooses to leave it empty (e.g. inserting "not specified" into the address field). Ensures that fields that are required by the gateway are populated with an acceptable value.

(g) Alpha / Alpha– Numeric / Numeric Only: Issues an error message if the user enters the wrong data format. Reduces the risk of data entry errors.

(h) All Caps / Lower Case: Issues an error message if the user enters the wrong data format. Reduces the risk of data entry errors.

Step 3: Integrate to the Payment Gateway

Integrating the payment gateway to an application (such as a website or billing system) can vary greatly in terms of the complexity involved. In general, the following steps will apply:

1. Integration Method Selection

Most payment gateways have many integration methods depending on the type of interface sending the transaction, where that interface is hosted (i.e. operating system) and how complex the integration is. For example, some implementations may use a simple form post to the payment gateway, whereas more complex ones may require the use of an API.

Mapping out the data flows and system requirements for the application against the payment gateway integration methods will help determine the appropriate method. Generally speaking, use of an API provided by the gateway provides the most flexibility, but can be more complicated to integrate and maintain.

2. Data Field Posting Configuration

In this step, you should ensure that you are passing all of the required and optional fields from the interface to the gateway using the correct syntax. Required fields are the minimum ones necessary to complete the transaction. Optional fields contain useful supplementary data. They must be posted to the gateway using the syntax dictated by your gateway provider.

3. Response File Interpretation

Once the fields are posted, testing must be done on the response mechanism(s) to ensure that they correctly interpret all possible response outcomes from the payment gateway (e.g. approvals, declines, system errors, etc.). This involves simulating the various outcomes and making sure that the correct response values are mapped to the variables, and that everything displays properly in the response pages and email templates (if available).

In some cases, the payment gateway may actually display the variable name that has been used for a specific data field in the response page (e.g. if you use "CustName" as the variable, this may be what is displayed to the user in the response page, email, etc.). Wherever possible, try to use "customer–friendly" variable names rather than cryptic ones to avoid confusion on the part of the user.

4. Fail Safe Development

Put fail–safes in place to compensate for internet connectivity failures that might result in the customer

not getting a payment response from the gateway. This involves working through the logic of how a customer processes a transaction and receives notification of the transaction to identify possible points of failure (e.g. what would happen if Internet connectivity failure occurred immediately after a customer posted a transaction to the payment gateway but before the response page was returned? How does the customer know what the status of the transaction is?).

Fail-safes can take many forms. EasyPay123, for example, has a Get Transaction Status feature that allows the application to query Skipjack based on unique order number to determine (a) whether the transaction is in the system, and (b) if it, what its status is (e.g. approved, declined, settled, etc.). Once the status is known, the application can be programmed to determine the correct course of action to take automatically.

5. Third Party Linkages

In some instances, there may be a requirement to pass transaction data to a third party application, either at the time the data is posted to the payment gateway, or using the results returned from the payment gateway after the transaction is processed. There are a number of ways to do this, either by having the gateway post to a specific location, or by extracting and formatting the data for import into the third party application.

Talk to your payment gateway provider to determine the best method of integrating to the application.

Step 4: Test

Testing can be accomplished during the development stage by using the payment gateway's development environment (if one is available) and a test credit card. This allows the developer and the user to ensure that the application functions in the manner that it should. Development testing only provides assurance of the functional parts of the application, however. To ensure that funds are approved and settled properly, testing should also be performed with the application on the "live" platform before making it available for general use. We recommend the following:

1. Using a live credit card, authorize a transaction in the amount of \$1.00
2. Ensure that the transaction is successfully approved, and that:
 - (a) The response page is displayed and contains all of the necessary information
 - (b) The email receipt is received and contains all of the necessary information (if an email receipt is being used)
3. After 24–48 hours (depending upon the merchant bank), check the merchant bank account to determine if the funds from the authorization were successfully deposited.
4. Verify that the merchant name appearing on the live credit card statement used in Step 1 appears as expected (this is defined during the merchant account application process, and is typically a derivative of the "doing business as" name on the application).

Step 5: Communicate

Chances are that you built your payment application because it helped you streamline your operations,

Designing and Building Payment Applications

reduce your costs, increase your sales, improve customer service, or all of the above. If your customers/users are not aware that it exists, your success at achieving your objectives may be reduced. You will need to get the word out that your application exists. This could include advertising or leveraging search engines in the case of online retail, or a link to your payment interface in your emails and invoices in the case of bill payment applications. Many methods of communication exist, including advertising, affiliate programs, "viral marketing" efforts, search engine submissions, and others. The Internet is an excellent resource to research these various options. We also have an article, "Gaining Visibility for Your Website", that highlights a few of these different methods. You should also use your offline marketing efforts and collateral material to make customers aware of your application.

Don't ignore your employees when sending out communications. They should be aware of the value and implications of the new payment application, since it may affect how they do their job or interact with customers. Sending an internal bulletin to them describing the application, its value, and how it may impact them, will go a long way toward avoiding confusion once the application launches.

Design and Development Tips

- * Data validation and required field coding will dramatically reduce end–user generated errors and dissatisfaction by the end user.
- * Design and test for all browser types.
- * Test, test, test to ensure that all elements of the system are functioning properly. Mapping out the data flows and developing an exhaustive test protocol to follow will ensure that most issues will be identified and tested prior to launch.

Summary

A complete discussion of how to integrate a payment gateway to an application is beyond the scope of this document. There are, however, basic steps that should be followed regardless of the nature of the integration. If you are a developer looking to integrate a payment gateway, your gateway provider should provide you with the documentation, scripts, software modules and support to make your task easier. If not, then you may wish to look at alternate gateway providers. If you are a merchant that is engaging a developer to do this work for you, you may find value in using the points in this article to ensure that your developer is following a comprehensive approach that will result in an error–free, customer–centric solution.

About EasyPay123

EasyPay123 is a leading supplier of payment processing solutions to businesses across North America. Offering world–class solutions at affordable prices, EasyPay123 helps merchants simplify the process of acquiring, launching and using payment applications to improve the way they do business. Visit us at www.EasyPay123.com.

For a description of some of the e–commerce terms used in this article, please visit our online glossary at www.EasyPay123.com.

If you found this article helpful, you may wish to request one or more of the other articles in the EasyPay123 series by visiting our website. Articles in this series include:

- Understanding E-commerce Transactions
- How Transactions are Processed
- Getting Merchant Accounts
- How to Design and Build Payment Applications
- How to Choose a Payment Gateway
- How to Pick a Shopping Cart
- Preventing Online Fraud
- Gaining Visibility for Your Website
- Understanding Wireless Payments

© 2003 EasyPay123. All Rights Reserved. This article may not be copied, reprinted, published, translated, hosted, or otherwise distributed by any means without explicit written permission from EasyPay123.

Robert Levings is President of EasyPay123, a leading provider of online payment solutions to merchants across North America.

Advanced Hosting for the Mission Critical Web Presence

By Vincent Bezzina

As companies that conduct their business online have proliferated, hosting of those websites that provide the primary point of contact for transaction of business has become highly mission critical; having these sites go offline or perform poorly not only loses revenue but also detracts from the company's image and loses customer loyalty.

Companies such as betting and online gaming have even more stringent requirements because they need to provide a guaranteed response in near real time. Sports betting events also have the effect of crowding all the business within condensed time windows.

This level of hosting goes far beyond the simple provision of bandwidth and the quality of the server matters not only in terms of its CPU's processing abilities, quantity of RAM and hard disk space but also the durability of its components, the power supply available and the ability of fan/s to disperse the heat generated by a server running 24/7 under possibly very heavy loads.

With an application designed for use internally within a company, one can always put a cap on the maximum number of people that could be using the application. With the internet this number can be unpredictable, or if measurable through registration, can grow large very quickly. The internet is a new operating regime, not only in terms of the security issues it presents but also in the scale of operations, and this requires a new way of thinking when designing applications and the hardware architectures that host them.

The traditional approach is to scale up vertically, increasing the bandwidth, CPU/s' speeds, memory and so forth. There is a limit, however, to how far this can be taken and with so much depending on such a concentration of resources, a failure is nothing less than catastrophic. The answer is to achieve scalability horizontally with a distributed architecture. This architecture not only allows increased scalability, but also creates reliance to faults and failures within the system. What's more, this model is inherently suitable to most operations and services offered over the internet which are in themselves quite simple, but that there is just too many of them.

This situation is akin to how humans organize themselves to accomplish very large workloads; there comes a time when one person, no matter how hardworking and clever will not be able to cope. At that stage the tasks will be split among many people doing exactly the same task and yet coordinating their activities. Imagine if you will, people flooding into the premises of a bank or payment office. Many cashiers wait in booths doing exactly the same job, overseen by managers and perhaps a helper guiding the queues. The architecture and layout of the building hosting these activities, is itself designed to allow a smooth flow of people.

Hosting a distributed architecture is more complex than a traditional centralized system where everything happens in one place. Parallel events need to be coordinated so that they work as a seamless whole and transactional control assumes a critical role. Hardware setup and middleware software need to be designed much in the same way as a purpose-built building, and layers of middle

management in the organization would be in place for a human organization. The applications themselves should be aware that they are running in a distributed environment and be able to both benefit and not obstruct this environment.

The key to a successful internet presence stems from both an understanding of the nature of the internet and the tools that are now available to build up this success. The internet has indeed come a long way.

Endeavour has evolved its hosting services starting from its own demanding requirements for hosting its Internet Payment Gateways. Since then, Endeavour has been offering advanced hosting services to serve clients around the world. Real time backups, applications designed for distributed architectures, geographically distributed resources such as databases and fail-safe architectures are at the heart of these services.

Mr. Vincent Bezzina
(B. Eng. Hons.).

Mr. Bezzina has had a distinguished career across a number of industries, always in the IT field. Endeavour has delivered IT, consultancy and training services to major blue chip companies in 32 countries across the globe. The name of the company, Endeavour, espouses Mr. Bezzina's own philosophy

Advanced Hosting for the Mission Critical Web Presence
Bad Credit Auto Financing – 3 Ways To Get Approved More Easily

Using professional XP style icons in successful application development
Getting A Credit Card In An Instant
Beware of Balloon Mortgages

Profitable Crafts Vol 3
Profit Gadget
Free List Pro
Software Index
Secret Copy Writer



This Free E-Book has been brought to you by Natural-Aging.com.

100% Effective Natural Hormone Treatment
Menopause, Andropause And Other Hormone Imbalances
Impair Healthy Healing In People Over The Age Of 30!