Come implementare un pagamento online: chiarirsi le idee (2/6)

Nel precedente articolo abbiamo preparato tutto quanto ci servirà per attuare la nostra procedura di pagamento. Dunque:

- L'ambiente di simulazione con relativi account
- La pagina con il bottone paga adesso
- La tabella utenti del database

Ora dovremo intercettare la notifica di pagamento che PayPal ci invia al termine della transazione (IPN - Instant Payment Notification).

Definire l'url al quale inviare la notifica

Il primo passo da compiere é quello di indicare a PayPal l**a pagina alla quale inviare la notifica.** Logghiamoci dunque a Sandbox, selezioniamo l'account *admin* e logghiamoci a questo account. Clicchiamo su *profilo* ed in seguito su *Preferenze per la notifica immediata di pagamento* ed infine sul bottone scegli *impostazioni IPN*.
Scegliamo *ricevi messaggi IPN* ed inseriamo nel campo l'url al quale dovrà essere inviata la notifica, qualcosa tipo *http://www.tuosito.com/lettoreIPN.php.* Naturalmente bisognerà lavorare online, quindi scegli una cartella su un tuo hosting.

A questo punto possiamo salvare. Ora, ad ogni pagamento, PayPal invierà a questa pagina una serie di informazioni che noi dovremo intercettare, verificare ed elaborare.
Già che siamo nell'account *admin*, vediamo un'altra cosa: clicchiamo su *profilo* quindi su *informazioni commerciali.* Accanto al *nome di conttatto* clicchiamo su *modifica* e quindi scegli amo *Modifica della ragione sociale (nome dell'azienda).*
A questo punto possiamo inserire il nome che desideriamo far apparire in testata nella pagina del pagamento, io inserirò Your Inspiration Images.

Introduzione a IPN listener

Quello che dovremo implementare ora è, come detto, un file in grado di intercettare, verificare ed elaborare la notifica che PayPal invierà; questo file è tecnicamente chiamato **IPN listener**.
Tutti i parametri relativi alla transazione saranno inviati a questo file con il metodo POST.
Dunque, se vorremo sapere ad esempio il cognome della persona che ha pagato il servizio, lo troveremo in questo modo

$_POST['last_name']

Ma prima di buttarci a capofitto nell'elaborazione di questi dati, **dobbiamo assolutamente**

procedere a delle verifiche.
Prima di considerare la questione conclusa e quindi passare all'attivazione dell'account dell'utente dovremo verificare:

- **La legittimità della notifica**: chiunque può inviare dati tramite POST, dunque dobbiamo essere certi che la notifica arrivi veramente da PayPal e vedremo come.
- **La corrispondenza prodotto - costo**: quanto ha pagato l'utente deve essere corrispondente al costo del prodotto o del servizio.
- **Lo stato del pagamento deve essere "Completed"**: la notifica potrebbe arrivarci con il parametro *payment_status* con valore ad esempio **Pending**. Questo significherebbe che il pagamento non è ancora terminato ma è in attesa di essere processato. Può dipendere da molti fattori ed il tempo necessario potrebbe essere di alcuni secondi come pure di ore. Non ci interessa, noi procederemo con l'attivazione dell'account solo quando il pagamento risulterà completo. Nota che ad ogni modifica del *payment_status* PayPal invia una nuova notifica.
- **La transazione non deve già essere stata processata**: verificheremo che l'id della transazione non sia già presente nel database, come ho già spiegato nell'articolo precedente.
- **L'email iniviato nella notifica deve essere il mio email primario di PayPal.** Per farlo dovremo leggere il contenuto del parametro *receiver_email* e confrontarlo con il nostro email primario PayPal (vedremo come ricavare questo dato).

Solo quando questi controlli (che sono fortemente raccomandati da PayPal) avranno dato esito positivo, potremo procedere con:

- La creazione del nuovo account.
- L'invio dei dati di autenticazione all'utente.

Queste sono le funzioni che dovrà svolgere il nostro listener che ora inizieremo a progettare.

Progettare il listener

Ho pensato che nei miei tutorial ti presento sempre delle classi (sono un fanatico della programmazione ad oggetti) ma non ho mai presentato il ragionamento che dovrebbe stare alla base dell'utilizzo di questo paradigma. Infatti non si tratta unicamente di scrivere il codice in un modo diverso, ma di pensare e progettare in modo diverso, ad oggetti.
Dunque prima di buttarci nel codice dovremo analizzare alcuni punti:

- Di cosa avrò bisogno: quali funzionalità saranno necessarie, che metodi dovrò implementare.
- Quali di queste funzionalità potrò riutilizzare in altri progetti: iniziando dunque a pensare ad un possibile meccanismo di estensioni.

- Come queste funzionalità potranno interagire.

Iniziamo dunque a definire i metodi che ci serviranno decidendone già anche il nome.

isVerifiedIPN: con questo metodo verificheremo l'attendibilità della notifica.
sendReport: se il metodo isVerifiedIPN dovesse dare esito negativo, é probabile che si tratti di qualcuno che tenta di fare il furbo. Ma potrebbe anche trattarsi di un qualche errore del server, una lentezza. Il problema é che al momento dell'invio della notifica, l'utente ha già fatto il pagamento. Se dovesse verificarsi questa eventualità, facciamo in modo che il listener ci invii un email contenente i dati della transazione. In questo modo sarà più semplice verificare cosa é successo(l'ipotesi é molto remota comunque).
isVerifiedAmmount: il metodo che utilizzeremo per verificare l'effettiva corrispondenza tra quanto pagato ed il servizio acquistato.
isCompleted: questo metodo si occuperà di verificare se lo stato del pagamento é "Completed".
isPrimaryEmail: verifica se l'email del ricevente sia l'email primario su PayPal.
isNotProcessed: questo metodo verifica che la transazione non sia già stata processata
isReadyTransaction: questo metodo verifica che tutti i controlli (implementati con i metodi visti fino ad ora) abbiano dato esito positivo.

Fin qui abbiamo pensato ai metodi che ci serviranno sempre. Dunque questi metodi andranno a costituire la classe parent (la classe di base che dovremo poi estendere), mentre quelli che di volta in volta ci serviranno nello specifico li andremo ad implementare nell'estensioni di questa classe. Ma... c'é un ma.

Alcuni di questi metodi dovranno essere implementati in modi diversi a dipendenza delle circostanze, vediamo quali:
isVerifiedAmmount: nel nostro caso abbiamo un solo prodotto (l'accesso al sito) ed un solo costo. Ma potremmo vendere diversi prodotti con la conseguenza che questo metodo dovrà essere implementato in modo diverso.
isNotProcessed: anche qui non conosciamo l'implementazione a priori. Nel nostro caso salviamo l'id della transazione nella tabella utenti, ma potremmo decidere di creare una tabella apposita. Inoltre non conosciamo il nome dei campi che potrebbero essere utilizzati. Quindi anche questo metodo sappiamo che dovrà esserci ma non possiamo sapere come implementarlo.

Quindi nella serie di metodi dei quali avremo sempre bisogno e che di conseguenza faranno parte della classe parent, ve ne sono due che non sappiamo come implementare.
Tra l'altro anche questi due metodi insieme agli altri ci serviranno nel metodo isReadyTransaction, dove verifichiamo l'esito positivo di tutti i controlli. Allora come fare?

Le classi ed i metodi astratti

Ci troviamo di fronte al classico caso dove l'utilizzo dell'astrazione renderà il nostro codice più

elegante e più aderente alla prassi della programmazione ad oggetti. Dunque vediamo cosa intendiamo per classe e metodo astratto.

Se dichiaro una **classe astratta** (tramite la keyword abstract), questa classe avrà due fondamentali caratteristiche:

1. Dovrà obbligatoriamente essere estesa (tentare di istanziarla produrrà un errore).
2. Potrà contenere dei metodi astratti.

Un **metodo astratto** ha due proprietà fondamentali:

1. Dovrà essere implementato nell'estensione della classe.
2. Potrà comunque essere utilizzato nella classe parent in quanto dichiarandolo come astratto, ci siamo implicitamente impegnati a definirlo.

Dunque definiremo la classe parent come astratta. Al suo interno definiremo i metodi isVerifiedIPN, sendReport, isCompleted, isPrimaryEmail e isReadyTaransaction. I metodi isVerifiedAmmount e isNotProcessed li definiremo come astratti (vedremo in che modo).

Estendere la classe

A questo punto disponiamo di una classe sufficientemente generale e robusta; possiamo quindi pensare alle operazioni specifiche che dovremo implementare ovvero i metodi della classe child.

Naturalmente, come prima cosa, dovremo implementare i due metodi astratti. In seguito dovremo prevedere i seguenti metodi:

dbConnect: ci fornirà la connessione al database;
getRandPassword: si occuperà di fornirci una password casuale per l'attivazione dell'account;
sendLoginData: questo metodo invierà l'email con i dati di autenticazione;
insertNewUser: inserirà il nuovo utente nel database se isReadyTransaction da esito positivo.

Conclusione

In questo articolo abbiamo gettato le basi teoriche e conettuali per realizzare il listener che ci permetterà di intercettare ed eleborare la notifica di pagamento e quindi di fornire l'instant access agli utenti.
Nel prossimo articolo passeremo alla pratica realizzando la classe INPListener, ovvero la classe parent responsabile di eseguire i controlli necessari.

Allora, sei sempre più curioso di sapere come andrà a finire?

Articoli di questa guida