# Invoice Application

Alan Mulligan

It’s been just over two week but I think between myself and my mentors David and Thomas we have finally nailed down exactly what the project is going to entail.

The application I purpose to create is a desktop invoicing application. The user will have an F# windows form which they will use to select criteria (DateFrom, DateTo, InvoiceCurrency, Merchant and ProfitMargin). The user will then click the generate invoice button and the data will be serialized and sent to the server via ZeroMQ as a byte array.

The ZeroMQ process will be made of a request/reply pattern (The desktop application will be the request application and the reply application/server will be an F# console application). Each application will have a ZeroMQ socket created within them, these sockets which are like tcp socket are then programmed to be a certain type of socket, in our case a request and a reply socket as metioned above. But many more types of socket are available to create different patterns. To send data across these sockets it will have to be serialized before ZeroMQ encodes it to a byte array for transport. There are different types of serialization like xml and Bson but I have chosen Json. When the reply socket receives the data from the request socket it will have to be decode from the byte array and then de-serialized.

One of the extremely useful things that F# has is type provider, again there is several types. The Json Type Provider provides statically typed access to Json documents. It takes a sample document as an input (or a document containing a Json array of samples). The generated type can then be used to read files with the same structure. After using this method to access the data which has been entered on the UI and transported across ZeroMQ I will plug these values into a query and bring back all results from the database (PostgreSQL).

I have been supplied with actual data from Continuum database to process during the project, this contain over 80,000 records. The data I have its covers six different merchants. This means each query by merchant, which all queries will be should be about 16,000 records. At the moment I have only three tables the transaction table with all the records and a table with exchange rates. I have written a little F# application to request exchange rates for different currencies. These will be saved for each day corresponding to a date in the transaction table. And a currency table which is the parent of the rates table.

For accessing the database I am going to implement the Data Access Object Pattern. Data Access Object Pattern or DAO pattern is used to separate low level data accessing API or operations from high level business services.

As for the calculations on the data from the database these are going to be in Linq instead of using queries to the database. The only queries to the database will be by merchant. All calculations will be done within the server application itself.

An Invoice will also generated, the specs of this invoice are yet finalized but I would imagine it will be a text file with (date from, date to, total refunds, net total, profit margin, total owed to Continuum). These files will then be archived in the database again this is not full decided at this stage.