# Bookstore Design Pattern Report

### Introduction

This report will explain the patterns I used for my bookstore application, why I chose them and what role each participant class played. I chose to do this assignment using Java, MySQL and the Struts2 Framework. I chose these because I felt I have the most knowledge in these technologies as I am using them for my final year project.

The patterns I thought would suit this application were:

* MVC Pattern
* Singleton Pattern
* Data Access Object Pattern
* Transfer Object Pattern
* Intercepting Filter Pattern
* Strategy Pattern

### Model View Controller

The MVC Pattern is the Model View Controller Pattern. I chose to use the Struts2 MVC Architecture for my bookstore. Model represents the underlying, logical structure of data in a software application and the high-level class associated with it. In my struts2 application they are the pojo classes i.e. category, customer and book. View is a collection of classes representing the elements in the user interface. In Struts2, I have used JSP pages to represent my view. Controller represents the classes connecting the model and the view, and is used to communicate between classes in the model and view. These are my actions classes i.e. BookAction, PaymentAction, RegisterAction and so on. They take in the data the user enters in and do something to it.

### Singleton Pattern

This pattern involves a single class which is responsible to create its own object while making sure that only single object get created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class. I used the singleton pattern to create the ConnectionFactory class which makes a connection to the database. It only creates one instance of it so my ShopCartDAO and AdminDAO classes can just call ConnectionFactory.getInstance().getConnection(); to gain access into the database without having to have a lot of repeat code in both classes.

### Data Access Object

The data access object encapsulates all access to the data source. The DAO manages the connection with the data source to obtain and store data. My ShopcartDAO holds on the code for where I access MySQL database. I also have an AdminDAO which covers all the functionality that an admin needs to read from the relevant tables.

### Transfer Object Pattern

The Transfer Object pattern is used when we want to pass data with multiple attributes in one shot from client to server. Transfer Object is a simple POJO class having getter/setter methods and is serializable so that it can be transferred over the network i.e. my book, category and customer classes. It does not have any behaviour. Server Side business class normally fetches data from the database and fills the POJO and send it to the client or pass it by value. For client, transfer object is read-only. Client can create its own transfer object and pass it to server to update values in database in one shot. This is done by all my action classes.

### Intercepting Filter Patter

The intercepting filter design pattern is used when we want to do some pre-processing / post-processing with request or response of the application. I use my ShopCart Session Tracker class to do this. Filters are defined and applied on the request before passing the request to actual target application. Filters can do the authentication/ authorization/ logging or tracking of request and then pass the requests to corresponding handlers. In this case I am using it to keep track of all the requests.

### Strategy Pattern

A Strategy defines a set of algorithms that can be used interchangeably. It is used to change class behaviour at run time. I tried to use the strategy pattern to allow for the different payments methods that could be used at checkout i.e. cash or card.

### The Application Design

Customers can register and log in to their own accounts. They can then choose the category of book that they want. They will then be presented with a list of books in that category. They can choose one and proceed to checkout. From here they can choose to go back and add more books or they can proceed with payment. Here they have the option for cash or card. If card is chosen they will enter their card number and press submit and their bill will appear. If they choose cash they will be taken directly to their bill. Books will then disappear from the product list. Admins can log on and add books, update books or delete books.

I have included MySQL database files so that you can access them. The admin login is admin and the password is also admin.

### GitHub Account

<https://github.com/alisonf92/bookstore>