Design Patterns for Web Applications

Leeds Metropolitan University
Sophie M Greene
C7071971

Table of Contents

1 Introduction 1
1.1 Overview 1
2 Review of web technologies 1
2.1 Web Application Framework 1
2.2 PEAR (PHP Extension and Application Repository) 1
2.3 Enterprise Applications 1
2.4 Design Pattern 2
2.5 Design patterns categories 2
2.5.1 Command Pattern 3
2.5.2 Singleton Pattern 3
2.5.3 Registry Pattern 3
2.5.4 Front Controller Pattern 3
2.5.5 Database Design Patterns 4
2.6 Object oriented concepts and tools 4
3 File Share Web Application 4
3.1 Overview 4
3.2 Implementation 6
3.2.1 PHP 6
3.2.2 J2EE 7
3.3 Conclusions 8
4 References 9

1 Introduction

1.1 Overview

This report will discuss some current issues in web software engineering. It will also illustrate and evaluate the development of a file sharing website. The report will also draw a comparison of two different implementations of the website application using object oriented-based design patterns in two different programming languages; PHP and Java (Java 2 Enterprise Edition J2EE).

2 Review of web technologies

2.1 Web Application Framework

A Web application framework is a set of software tools that enable the creation of web applications which feature large amounts of functionality and dynamic content. Frameworks have a large amount of pre-written software which can be amended or used as required.

2.2 PEAR (PHP Extension and Application Repository)

"PEAR is a repository of quality-controlled PHP packages that extend the functionality of PHP. It is also a client-server mechanism for distributing and installing packages and for managing inter-package dependencies."(Zandstra, 2010, p.323)It is stored centrally and is used by PHP programmers to promote code reusability.

2.3 Enterprise Applications

Enterprise applications are designed to implement independent operations of several distinct tiers (separation of concerns).

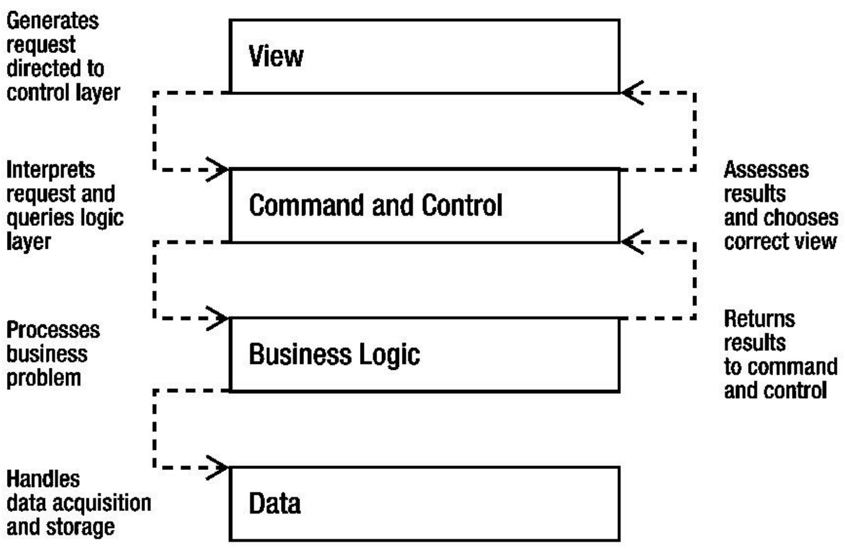
Generates request directed to control layer
View
Interprets request and queries logic layer
Command and Control
Assesses results and chooses correct view
Processes business problem
Business Logic
Returns results to command and control
Handles data acquisition and storage
Data

Figure 1(The layers or tiers in a typical enterprise system, Zandstra 2010, p.223)

- The view layer contains the interface that a system's users actually see and interact with. It is responsible for presenting the results of a user's request and providing the mechanism by which the next request can be made to the system.

- The command and control layer processes the request from the user. Based on this analysis, it delegates to the business logic layer any processing required in order to fulfil the request. It then chooses which view is best suited to present the results to the user. In practice, this and the view layer are often combined into a single presentation layer. Even so, the role of display should be strictly separated from those of request handling and business logic invocation.
- The business logic layer is responsible for seeing to the business of a request. It performs any required calculations and marshals the resulting data.
- The data layer insulates the rest of the system from the mechanics of saving and acquiring persistent information. In some systems, the command and control layer uses the data layer to acquire the business objects with which it needs to work. In other systems, the data layer is hidden as far as possible.

2.4 Design Pattern

"Design patterns are general reusable solutions to recurring problems in software development. They capture well proven experiences of expert software developers regarding commonly occurring problems and their solutions. Pattern based design methodology is used to build software systems by instantiating and composing existing design patterns. A key advantage of this methodology is reusability of existing solutions and so reduced cost."(Singh & Chaudhary, 2009, p.283)

A pattern is described by its name, the problem it provides solution for, the solution and any consequences of implementing that solution

Design Patterns offer a lot of advantages through defining problems and defining tried and tested solutions, being language independent, facilitating clearer communication between programmers by putting names on specific techniques, being designed for collaboration, i.e. more than one pattern can be applied to solve different parts of the design problem and finally, promoting good design practice by applying principles of object oriented design.

2.5 Design patterns categories

Design patterns can be categorised into the following

- Object Generating Patterns: concerned with the instantiation of objects. When working with abstract parent classes in the application's design, strategies for instantiating objects from concrete subclasses need to be developed. Examples: Singleton.
- Patterns for Organizing Objects and Classes: help developers to organize the compositional relationships of objects in the application by identifying ways of combining objects and classes.
- Task-Oriented Patterns: describe the mechanisms by which classes and objects cooperate to achieve objectives of the application. E.g. command patterns
- Enterprise Patterns: describe typical Internet programming problems and solutions. These patterns mainly deal with presentation, and application logic.
- Database Patterns: help with storing and retrieving data and with mapping objects to and from databases.

2.5.1 Command Pattern

In order to achieve independence of code, decoupling is paramount in object oriented programming. Tightly coupled code requires changes whenever the code is reused. Using the Command Pattern encourages the separation of concerns and hence used in enterprise applications which result in an organised application that is easy to expand.

2.5.2 Singleton Pattern

Global variables tie classes into their context, undermining encapsulation and reusability. Global variables are unprotected and hence problematic. There is no protection against clashes with other global variables declared elsewhere. PHP does not issue a warning when global variables. However, global variables remain a temptation in order to give different classes, access to the same object.

2.5.2.1 The Problem

The Singleton pattern is designed to address global variables problem by being available to any object in the system. There should be no more than one object in play in the system. This means that object A can set a property in the object, and object B can retrieve the same property, without either one talking to the other directly (assuming both have access to the object).

2.5.2.2 Implementation

A solution can be created by asserting control over object instantiation. A class that cannot be instantiated from outside of itself is created by defining a private constructor. Access to the object is provided through a static method. Magic functions such as PHP's ___clone() and __wakeup() are used to enforce the singleton approach.

2.5.3 Registry Pattern

Enterprise systems are divided into layers, with each layer communicates with its neighbours only through tightly defined channels. This separation of tiers makes an application flexible. A registry pattern is useful in making some information available to all classes throughout the application (MO)

A registry is a class that provides access to data or objects via static methods or via instance methods on a singleton. Every object in a system, therefore, has access to these objects.

2.5.4 Front Controller Pattern

This pattern is very useful in large systems where flexibility is needed in managing many different views and commands.

PHP implementation of the front controller shows one of its most obvious weaknesses; initialisation is needed every time a request is handled.

2.5.5 Database Design Patterns

Data access abstraction is promoted in the MVC design architecture as part of the model part or the data layer. Separating the data access from the rest of the application is much desired in order to promote minimal changes to the system when data sources change. Singletons can be used to achieve that. A data access object is used as an interface between the data and the rest of the application.

2.6 Object oriented concepts and tools

- Decoupling is paramount in OOP in order to achieve independence of code or separation of concerns. Tightly coupled code requires changes whenever the code is reused
- Splitting the application into layers, separating application logic from presentation and data layers makes the application extendable and easy to manage.
- Coding to an interface, not to an implementation. Different implementations can be hidden behind the common interface defined in a superclass. Client code can then require an object of the superclass's type rather than that of an implementing class, unconcerned by the specific implementation it is actually getting, which promotes encapsulation.
- Composition versus inheritance: composition can make code more flexible, because objects can be combined to handle tasks dynamically in many more ways than in an inheritance hierarchy alone. This is due to the fact that composition relies on instantiating objects from within other objects. The problem with composition though, is that it makes the code less readable.
- Polymorphism offers great flexibility by the switching of concrete implementation at runtime through method overriding, overloading and dynamic or late binding.

All of the above techniques are used by design patterns to make the web development process easy, flexible and extendable.

3 File Share Web Application

3.1 Overview

The application has been developed using two different methods and two different IDEs. It is a file sharing application with user authentication and access rights capabilities. User groups are also supported to enable “admin” users to perform additional tasks using the application. Figure 8 shows a user case diagram and class diagram of the application

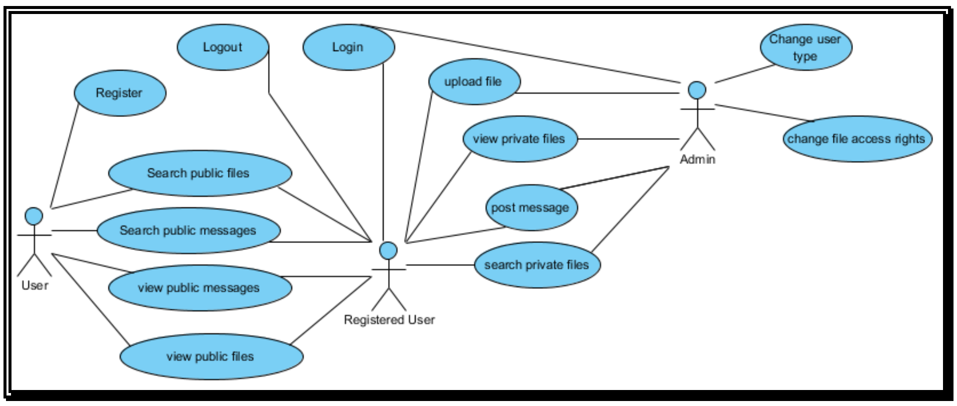

Figure 2(Use Case Diagram of Message-File Share System)

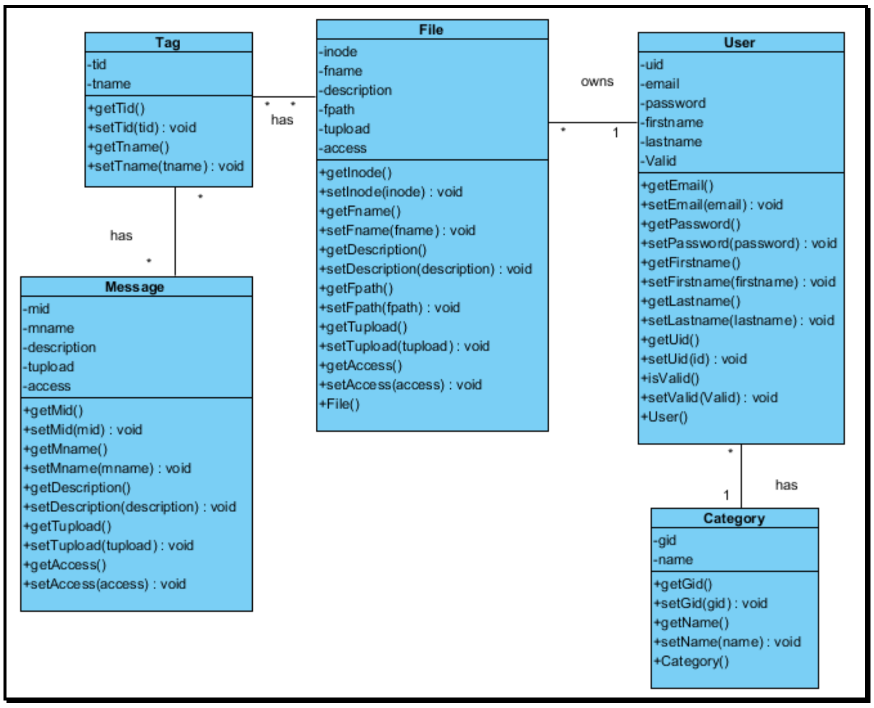

Figure 3(Class Diagram of File Share System, Data Objects)

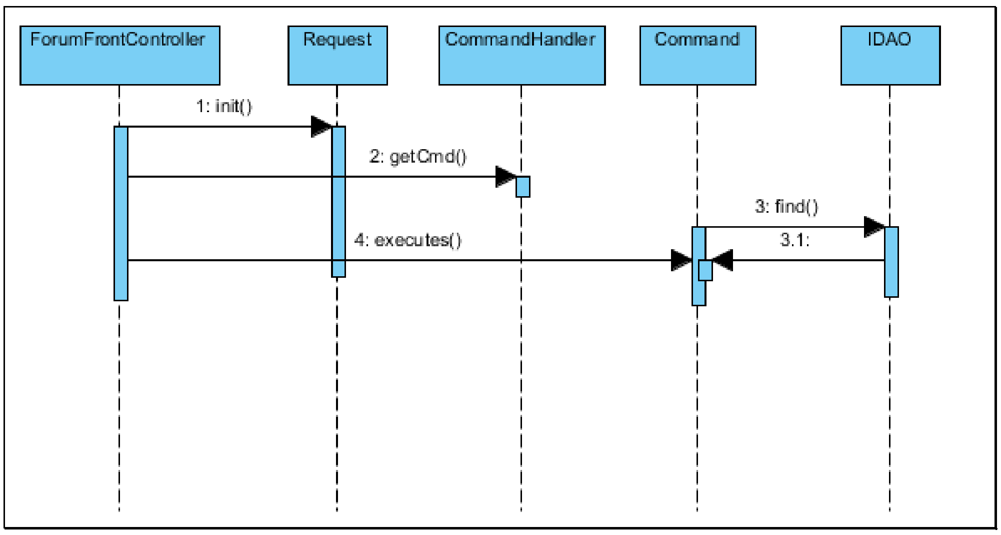

Figure 4(general sequence diagram of the system)

3.2 Implementation

3.2.1 PHP

The application is implemented in PHP by deploying four design patterns and applying an enterprise layering model (Model-View-Control)

The following class diagram shows the implementation of front controller and command design patterns.

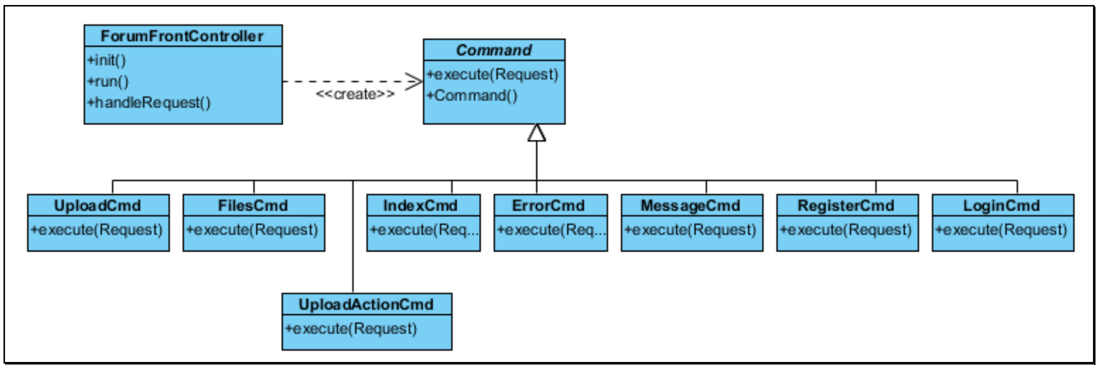

Figure 5(Class Diagram of Front Controller and Command Patterns)

Singleton pattern is used to control access to the database, the figure below shows its implementation

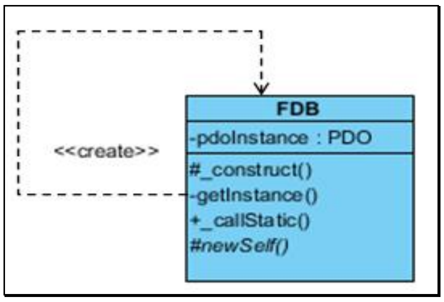

Figure 6(Singleton)

The data access object is used by the commands to connect to the database, the figure below shows a class diagram of the DAO. An interface IDAO is first declared then the implementation is userDAO which is application of OOP recommendation of encapsulation.

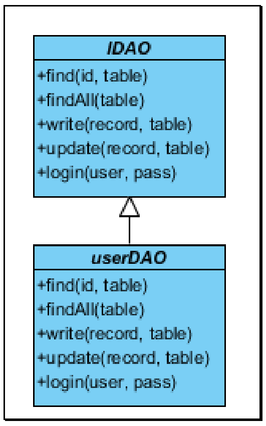

Figure 7(DAO)

3.2.2 J2EE

Eclipse IDE is used to develop a J2EE type application. The benefit of using this method over PHP is the embedded support for design patterns in J2EE.

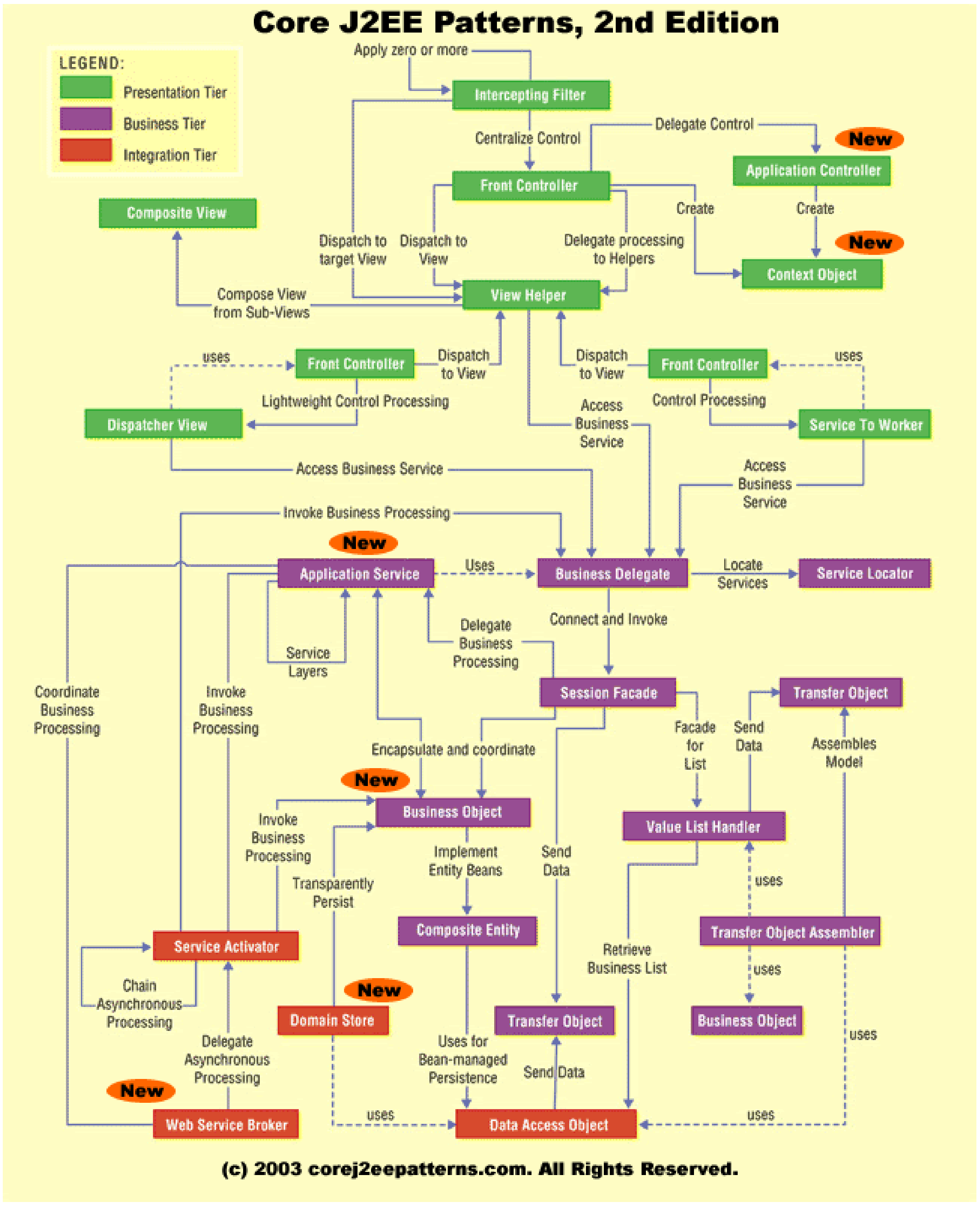

Figure 8(Core J2EE design pattern n.d.)

All of the patterns implemented in the PHP version are implemented in this version. The application takes advantages of the many design patterns implemented in the core J2EE design patterns shown in the figure above.

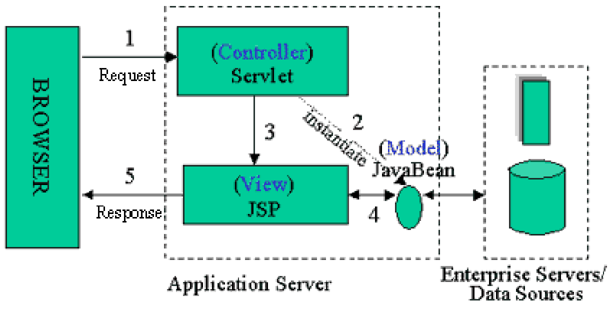

Figure 9 (JSP Model 2 architecture, JavaWorld, n.d)

The above figure shows the model followed for developing the file share website. JSP (Java Server Page) was used to implement the views and JAVA for the application's classes. The classes shown in figure 2 are the classes implemented to correspond to JavaBean on figure 9.

The intercepting filter is implemented to facilitate user authentication and unauthorised access prevention.

3.3 Conclusions

Many patterns apply to many object-capable languages with few or no implementation issues. This is not always the case. Some enterprise patterns work well in languages in which an application process continues to run between server requests such as Java. PHP initiates a new script execution for every request, which impacts on the implementation and operation of Front Controller which often requires some serious initialization time. This is fine when the initialization takes place once at application start-up but more often issue when it must take place for every request.

However, the huge flexibility in PHP can be very beneficiary to implementation of some patterns which require typecasting or dynamic overloading. There are many powerful tools through which flexible, extensible web applications can be implemented such as ZEND framework used widely in ecommerce, another example is the PEAR packages which use design patterns elegantly.

J2EE is a very powerful tool and needs a very solid understanding of object oriented issues and tools in order to be fully utilised.

4 References

1. “Web application framework” (2009) A Dictionary of the Internet. Darrel Ince. Oxford University Press. Oxford Reference Online. Oxford University Press. Leeds Metropolitan University. Available from:<http://www.oxfordreference.com/views/ENTRY.html?subview=Main&entry=t12.e4409>[Accessed 27 April 2011]
2. Apache Software Foundation (n.d.) **Apache Struts – Welcome**[Internet]. Available from: <http://struts.apache.org/> [Accessed 27 April 2011].

3. Core J2EE Patterns. Available at: http://www.corej2eepatterns.com/Patterns2ndEd/ [Accessed May 23, 2011].

4. Farley, J. & Crawford W. et. al. (2006) **Java Enterprise in a Nutshell (a Practical Guide).** 3rd Ed. United States of America, O’Reilly Media, Inc.
5. Fielding, R.T. & Taylor, R.N. (2002) Principled design of the modern Web architecture. **ACM Trans. Internet Technol.**, 2 (2), p.pp.115-150.

6. JavaWorld (n.d.)Server-side Java: Understanding JavaServer Pages Model 2 architecture - JavaWorld [Online image]. Available from: <http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html> [Accessed 27 April 2011].

7. Singh, P.B. & Chaudhary, B.D. (2009) Structural Formalization of Design-Pattern Based Software Design. In: **Third International Conference on Digital Society**. pp. 283-288.

8. Sun Developer Network, Java 2 Platform, Enterprise Edition (J2EE) Overview [Internet]. Available from: <http://java.sun.com/j2ee/appmodel.html> [Accessed 27 April 2011].

9. Zandstra, M., 2010. **PHP objects, patterns, and practice**, DE: Apress. Available at: http://www.dawsonera.com.ezproxy.leedsmet.ac.uk/depp/reader/protected/external/AbstractView/S9781430229261.