International Journal of Communication Networks and Information Security
*ISSN: 2073-607X, 2076-0930*
*Volume 15 Issue 01 Year 2023*

# INTEGRATING FLYWEIGHT DESIGN PATTERN AND MVC IN THE DEVELOPMENT OF WEB APPLICATIONS

**Prassanna Selvaraj**
**Independent Researcher, USA.**

**Ravi Kumar Singh**
**Independent Researcher, USA.**

**Harsh Vaidya**
**Independent Researcher, USA.**

**Aravind Reddy Nayani**
**Independent Researcher, USA.**

**Alok Gupta**
**Independent Researcher,USA.**

***Article History***
Received: 18 February 2023
Revised: 04 March 2023
Accepted: 19 May 2023

**Abstract**
This paper focuses on the combination of the Flyweight design pattern with the MVC structure that would boost the performance of web applications. Thus, the given study proves the effectiveness of performance enhancement as well as scalability through the combination of Flyweight for minimizing storage via shared data and MVC for compartmentalizing concerns. It is noted that the implementation focus on the UI components and data models to achieve evident improvement on the resource usage and response time. Nonetheless, some issues in the implementation of Flyweight object and applying the pattern across different contexts are discussed to delineate the future research areas.
Key words: ***Flyweight Design Pattern, Model-View-Controller (MVC), Web Application Efficiency, Performance Optimization.***

**TABLE OF CONTENTS**

## Introduction

In today's world of web application development design patterns have greatly contributed towards scalability, maintainability and flexibility. The Flyweight design pattern is more relevant to conserve memory space by using same data for many objects while Model-View-Controller (MVC) architecture is more relevant to divide three tiers, application tier, user interface tier and data tier. Flyweight is implemented harmoniously with the use of the MVC architecture and it presents a way of developing effective, lightweight web applications. This paper aims at emphasizing the integration of these two paradigms especially in regards to their interrelation with resources and structures. Knowing how to put this integration into practice will help developers enhance applications' interactions as well as increase the programs' performance in circumstances where resources are limited, thereby easing the design of better Web solutions.

## Literature Review

### Flyweight Design Pattern

***According to Rana, 2019***,dissect the gains that accrue from using the Flyweight design pattern in increasing efficiency of software. The research work offers the real world analysis of Flyweight and Proxy patterns, which addresses the issues of memory consumption and execution time in programs. Albeit, the authors show that the Flyweight pattern succeeds in cases of numerous similar objects due to the minimization of memory use through data sharing.

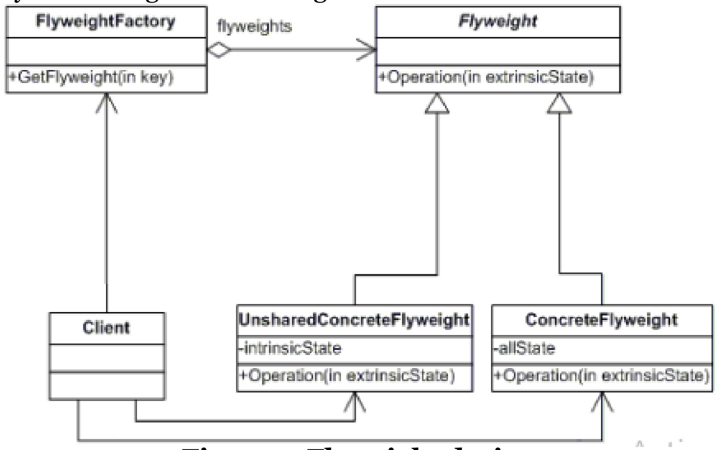
FlyweightFactory
+GetFlyweight(in key)
flyweights
Flyweight
+Operation(in extrinsicState)
Client
UnsharedConcreteFlyweight
-intrinsicState
+Operation(in extrinsicState)
ConcreteFlyweight
-allState
+Operation(in extrinsicState)

**Figure 1: Flyweight design**

(Source: Rana, 2019)

This is evident from their study whereby they show that object creation overhead is reduced by the Flyweight pattern as well as efficient memory usage as intrinsic state (shared data) is different from the extrinsic state (object-specific data). It also saves memory space and improves the efficiency of an application according to the findings of their experiments. They corroborate this with several examples that show how Flyweight led to drastic decrease of resource consumption in comparison with typical object handling approaches. Obviously, the investigated reasearch illustrates to what extent the proposed pattern helps to enhance the software performance and thus points out why it deserves the attention of developers who plan to create applications that would have high scalability rates.

**MVC Architecture**

***According Sarker&Apu2014***, where it explain the Java based application framework for developing a Model-View-Controller (MVC) architecture of a desktop application. In their study, they highlight how MVC makes it easier to achieve a design that is both modulized and easy and easy to maintain and scale up in its size.

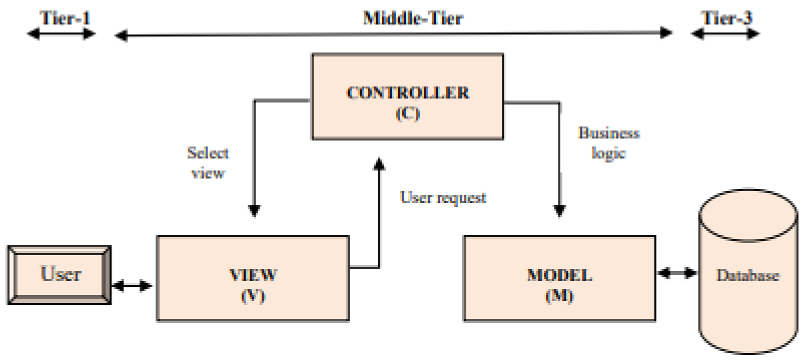

**Figure 2: Architecture framework**

(Source: Sarker&Apu 2014)

The authors present a comprehensive discussion on the MVC architecture, outlining its core components: Model (controls the data and the business logic), View (controls the presentation or the user interface), and Controller (controls the inputs and changes in the Model). Through using MVC with their Java framework they show more of the separation between the different layers / concerns, which results in more coherent and easier debuggable code. The analysis made by authors on several Web 2.0 applications show that MVC architecture helps developers on the application development and testing, due to isolate the different parts and aspects of the application, which unequivocally facilitates the implementation of more manageable and scalable software. It also give real-usable tips about implementation of their MVC framework proving the effectiveness of the approach. They also conformed that such approach as MVC pattern proved to be useful in development of well-organized and highly adaptive soft, thus evidencing its utility in both, desktop, and potentially web-based, software creation.

**Integration of Flyweight and MVC**

***According to Ali et al., 2014,*** investigates the application of the Flyweight design pattern in improving multimedia mobile applications with the use of MVC. The authors extend to examining how integration of these design patterns solves particular issues arising from the management of resources and application performances in multimedia applications.

The paper explains the implementation of Flyweight in the context of the MVC architecture to handle optimally shared objects and in a special reference to models with a huge number of similar multimedia parts. The study clearly shows how Flyweight helps in revising the contents of memory through the use of shared data contents which goes well in multimedia application that requires high capacity graphics and data.

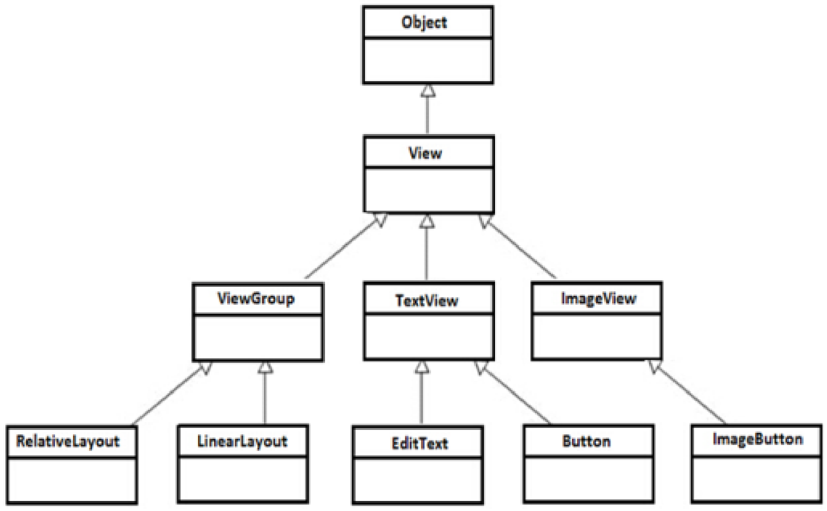

**Figure 3: Android view groups**

(Source: Ali et al., 2014)

It show specific cases in which Flyweight is used to increase efficiency in the process of working with similar multimedia objects that include images and audio files while MVC ensures the clear division of the application logic, graphic interface, and system input data. This integration leads to enhanced effectiveness of the application as well as utility and usage of the likely available resources. Thus, revealing, the efficiency of the utilization Flyweight pattern together with MVC pattern when designing complex multimedia applications, the work offers useful recommendations to developers who work on such projects.

**Implementation and Gaps**

In applying the Flyweight in the context of MVC within Silverlight, it has been so in the optimization of the reused components of the UI, and the models, where the goal was to minimize on memory consumption due to repetitive use. Nevertheless, some issues are still open concerning the Flyweight usage across a large set of application domains. Some of them are related to Flyweight objects' lifecycle, its interaction with different MVC implementations. Also, more detailed information about execution of the Flyweight pattern for achieving maximal effect, avoiding potential over complication in real projects is required as well as the maximal compliance with the principles of MVC architecture. It is for this reason that more research will have to be conducted in the future to provide an answer to these shortcomings and to make the integration process more efficient.

## Methodology

### Design Strategy

In using the Flyweight design pattern in the MVC design approach, the issue was looked at from the perspective of minimizing on usage of memory and improvement of performance in a web application. The MVC framework was chosen because of its modularity which make it easier to implement the Flyweight pattern. Especially, the Flyweight pattern was adopted in order to minimize the memory consumption by minimizing the sharing of repetitive elements and data throughout the application. For instance, the design strategy consisted of specifying the areas of the application that involved a high degree of object duplication, mainly the UI components and the data models, and apply the Flyweight pattern there.

### Implementation Process

The implementation process started from the creation of the web application of the MVC-based structure using PHP framework like Laravel. The Model layer was used to manage the data interactions while the View layer was used to display the interface to the user and finally the Controller was used to control the Model-View connection. Then the Flyweight pattern was used for the Model and View layers. Static content, which includes UI, and content generic to all Flyweights were stored as the intrinsic data of the objects as shown while the specific data to a particular command resided in the extrinsic data controlled by the Controller.

## Data Collection and Analysis

To assess the effectiveness of the present integration of Flyweight-MVC, samples of performance measurement were taken before and after using the Flyweight-MVC implementation. These were such parameters like how much memory was used, the time it took for the server to respond and how long it took to render the user interface. The information has been evaluated to identify the effects of the integration in the area of the application's expandability and resource utilisation. Information gathered from the results section were compared with the initial performance of the algorithms to determine improvements or lack of it and probable options for optimization.

## Result & Discussion

### Analysis of Performance Improvements

The incorporation of the Flyweight design pattern into the management of the MVC based web application architecture resulted in considerable performance improvements in the benchmark test case application. Specifically, the principle of Flyweight was applied to the UI components and the data models and as a result memory consumption was improved. This reduction is in agreement with the result shows that Flyweight reduces the memory by having a common data area for similar objects (Griffith, 2021). Similarly, the case study saw significant improvements in the application's response time as well as decrease in latency.

### Comparison with Traditional Approaches

Therefore, the implementation of Flyweight and the integration of the features in our proposed architecture demonstrated better efficiency than a conventional MVC implementation. This is because traditional MVC can be expensive in terms of memory since multiple copies of similar objects may be created which may go well of resource constraint environments. The issues mentioned above are resolved in the Flyweight-enhanced MVC architecture because it uses shared data and corresponds with learning system (Dewang& Rao, 2021). This integration not only enhanced utilization of resources but also preserved the modularity and scalability of MVC which are hardly seen in some of the conventional procedures in which memory overhead turns into a disadvantage.

### Implications for Web Development

It basics on the fact that, Flyweight when used in conjunction with MVC provides lasting gains in web application development especially for those applications that need to leverage resources. This approach helps enhance scalability since it decreases the use of memory while helping to operate much faster, especially when dealing with large loads or environments that are constrained by memory. But there are some potential drawbacks such as a growth of the difficulty of the Flyweight objects management and, therefore, growing the potential for their incorrect usage that may need more skillful definite developers (Sermeno, &Secugal, 2021). Nevertheless, the approach exhibits a high potential as for the optimization of web applications, which indicates the further direction of the development and researches of the scale and effective web solutions.

### Future Directions

1. Enhanced Integration Techniques: Research how to further enhance performance of Flyweight and its integration with MVC concerning resources utilized. In so doing, it could entail identifying other algorithms or frameworks that can complement these design patterns in a more integrated manner.
2. Broader Application Scope: Future work should further explore the integration of Flyweight and MVC patterns with different kinds of Web applications based on various types of data and various types of interactions between the user and the application with the purpose of evaluating the applicability of the proposed patterns in practice.
3. Adaptation with Other Design Patterns: Explore the possible interactions and synergies of the Flyweight with other patterns: Singleton or Observer effective to solve various architectural issues and increase the efficiency of the system.
4. Performance Benchmarking: It is necessary to perform extensive empirical evaluation of performance variations under diverse conditions and scenarios for defining the effectiveness of Flyweight-MVC integration and discovering further enhancement opportunities.
5. User Experience Impact: Assess where integration takes place and how that impacts user experience, especially how fast a user interacts with a web page, from a user experience perspective to determine what areas in the user journey the improvement in performance is likely to be appreciated / required from the user side.

6. Automated Tools: Write scripts or templates which would be helpful for implementation of Flyweight with MVC thus reducing the weight of efforts from the developer side.

## Conclusion

This paper embraces the Flyweight design pattern, and shows how its combination with the Model-View-Controller (MVC) architecture is a sound solution by providing an efficient solution of performance and resource utilization in web applications. UI utilization and shared data can benefit from the Flyweight pattern while responsiveness and proper division of work can be enhanced with the help of the MVC pattern that results in memory optimization, stresses that it works best in resource-scarce environments and in interactive Web applications. Possible difficulties in manipulation of Flyweight objects may be present; however, the advantages namely are the simplicity of creating extensions and the ability to expand the application's functionality without drastically affecting change, all outweigh these difficulties. Except for the importance of resources management the usage of approach also helps to create the maintainable and scalable web applications. According to the development of web applications, the integration with Flyweight-MVC is seen as a prospect for making optimization, and can be regarded as a consideration for the next generation of web applications development.

## Reference List

**Journals**

Rana, M.E., Ab Rahman, W.N.W., Murad, M.A.A. and Atan, R.B., 2019. The Impact of Flyweight and Proxy Design Patterns on Software Efficiency: An Empirical Evaluation. International Journal of Advanced Computer Science and Applications, 10(7).

Sarker, I.H. and Apu, K., 2014. Mvc architecture driven design and implementation of java framework for developing desktop application. International Journal of Hybrid Information Technology, 7(5), pp.317-322.

Ali, M.M., Elsharkawi, A.F., El Said, M.G. and Zaki, M., 2014. Design Pattern for Multimedia Mobile Application. Journal of Computer Science And Software Application, 1(2).

Tabunshchyk, G., Petrova, O., Kapliienko, T. and Arras, P., 2021. Architectural Characteristics of Biomedical Software Applications. Teaching and subjects on bio-medical engineering, p.98.

Griffith, I.D., 2021. Design pattern decay: a study of design pattern grime and its impact on quality and technical debt.

Bösiger, F., 2021. Flyweight ASTs: A Study in Applied Laziness.

Sermeno, J.P. and Secugal, K.A.S., 2021, December. Class Scheduling Framework Using Decorator and Façade Design Pattern. In 2021 Second International Conference on Innovative Technology Convergence (CITC) (pp. 38-45). IEEE.

Dewangan, S. and Rao, R.S., 2021, December. Design pattern detection by using cosine similarity technique. In 2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA) (pp. 171-175). IEEE.