T.C.

ALINBAŞ ÜNİVERSİTESİ

Electrical and Computer Engineering

MULTI-PLATFORM GENERIC DYNAMIC WEB PATTERNS

Mustafa Abdulkhudhur Jasim

Master of Science Thesis

Asst. Prof. Dr. Sefer Kurnaz

İstanbul 2019

MULTI-PLATFORM GENERIC DYNAMIC WEB PATTERNS

by

Mustafa Abdulkhudhur Jasim

Electrical and Computer Engineering

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2019

DEDICATION

I give my fruit to my dear father and dear mother who stood with me in the most difficult moments, without them reaching this stage ... and to my dear sister and brothers who always stood with me and were the greatest supporters of me ... and to the friends who stood with me in my studies in the most difficult situations. And my cousin Roida who encouraged me and stood with me in the worst of times and the worst moments ... and the supervisor who always gave me advice and steps in completing this research ... and to all my teachers ...

ACKNOWLEDGEMENTS

I wish to express my acknowledgements to my supervisor, Asst. Prof. Dr. Sefer Kurnaz who was abundantly helpful and offered invaluable support with his sincerity and belief in me.

ABSTRACT

MULTI-PLATFORM GENERIC DYNAMIC WEB PATTERNS

Jasim, Mustafa Abdulkhudhur

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Asst Prof. Dr. Sefer kurnaz

Date: March 2019

Pages: 67

It is complex and expensive to develop generic web-based software applications. Software solutions must understand each platform, technology and architecture's behavior and environment. The objective of the work is to produce a generic web pattern generator for three platforms of dynamic web developers: the Java EE, PHP Net. We have used the generic design architecture of the MVC (Model-View-Controller). Based on their specific technologies, the analysis of each layer within MVC can be analyzed. The lists of technologies being explored are as follows: Layers: Java EE: JSF and JSP .NET: Razor and Blade .NET; PHP: Twig and Blade; controllers layer: Java EE: Spring MVC .NET: .NET Framework. PHP: Laravel; Models Layers: Java EE: Hibernate and EclipseLink. We have developed a software application for student enrolment in courses to validate our proposal. The application has been implemented on the basis of all previously described platforms, technologies and architectures. A functional test to test the dynamic web functionality has been conducted. We have also conducted compliance tests to verify that the rules, specifications and structures have been followed by the specific platforms defined.

Keywords: Model View Controller, Web Application Generation, C#, Java, PHP.

ÖZET

ÇOKLU PLATFORM GENEL DİNAMİK WEB PATTERLERİ

Jasim, Mustafa Abdulkhudhur

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Danışman: Asst Prof. Dr. Sefer kurnaz

Tarih: March 2019

Sayfalar: 67

Genel web tabanlı yazılım uygulamaları geliştirmek karmaşık ve pahalıdır. Yazılım çözümleri her platformun, teknolojinin ve mimarinin davranış ve ortamını anlamalıdır. Çalışmanın amacı, dinamik web geliştiricilerinin üç platformu için genel bir web deseni üreteci üretmektir: Java EE, PHP Net. MVC'nin genel tasarım mimarisini (Model-View-Controller) kullandık. Özel teknolojilerine dayanarak, MVC içindeki her katmanın analizi analiz edilebilir. Araştırılmakta olan teknolojilerin listesi aşağıdaki gibidir: Katmanlar: Java EE: JSF ve JSP .NET: Razor ve Blade .NET; PHP: Dal ve Bıçak; denetleyiciler katmanı: Java EE: Bahar MVC .NET: .NET Framework. PHP: Laravel; Model Katmanları: Java EE: Hazırda Beklet ve EclipseLink. Teklifimizi doğrulamak için derslere öğrenci kaydı için bir yazılım geliştirdik. Uygulama daha önce tarif edilen tüm platformlar, teknolojiler ve mimariler temelinde gerçekleştirildi. Dinamik web işlevselliğini test etmek için fonksiyonel bir test yapılmıştır. Kuralların, özelliklerin ve yapıların tanımlanmış belirli platformlar tarafından takip edildiğini doğrulamak için uyumluluk testleri de yaptık.

Keywords: Model View Controller, Web Application Generation, C#, Java, PHP.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

1. INTRODUCTION

Web production is increasingly dynamic and different technologies have their own advantages simultaneously. A dynamic web development architecture is very important in addition to the technology. The architecture is the foundation for dynamically defining and shaping the Web. Model View Controller (MVC) is one of the most popular architectures. With the possibility to reuse, MVC strictly separates the application layer. The complexity of platforms, systèmes and different Web architectures, and huge technical costs. Programmers work hard to understand every platform's technology, architecture, methods, behaviour, environment and technology. On the other hand, programmers must produce dynamic web applications of high-quality applications in less time. There are many technical details that take time to understand and influence the overall development of software projects.

By removing the particulars and complexity of each platform and dynamic Web technology, abstraction can produce a model of invariable value. Generic design is the solution architecture or structures which define the general characteristics of the generic pattern's implementation behaviour.

The aim of this study is to develop an extensive pattern for the web developers of Java EE, NET and PHP. In order to improve productivity for programmers, the resulting platform design can be used on a wide range of technology platforms, while the programmers must transform technologies into dynamic web development.

What else has been examined in this thesis, the analysis, design and implementation of software applications / tools are then presented. We carried out a case study of the software application / tool's functionality and compliance. Finally, the research was finished.

It may be defined as electronic commerce when buying or selling goods on the internet. Electronic commerce, which has grown explosively in the internet use, has been one of the largest and rapidly growing Internet activities worldwide in the last 10 years. Forrester Research expects its online retail sales in the United States to reach $278.9 billion in 2015. Due to increased internet usage and technology improvements, the e-commerce face developed during the years, from the process to electronically e-transactions through eTransfer(EFT) to an entire

shopping experience for today's consumers. Service providers now have to offer a guaranteed level of service to their customers. In view of the size of the market and of its e-commerce reliance on companies, it was essential for e-commerce applications to develop highly available, scalable and low response times. The architecture and flexibility of E-Commerce systems have therefore been closely examined in recent years in connection with changing business needs, with increasing growth and ease of use. In this respect, applications that follow a multi-level architecture approach have become the norm for e-commerce companies. The whole system is essentially divided by the multilevel architecture into three distinct components: customer, application and data level. The customer level is the point of interaction between users and the system. At the core of the e-commerce system, the application level includes a business sense. Finally, the level of data is the level at which all relevant data are stored. The design, maintenance and scale of such systems are easier as components are separated.

We can add hardware on each machine or replicate a level to increase the number of resources available in each level to scale multi-stage architectures. Replication is often the preferred approach, because the availability of the system is cheaper and better. When we use replication, a load balancer is usually positioned in front of the replicated level, which functions as a contact point for the previous level. The data replication is not as common as that of the application because of the difficulty in managing a distributed database. Thus the system bottleneck often has a single database level. This particularly applies to e-commerce applications based on dynamics and data. The size and load of the database layer increases dramatically when the total numbers of system users increase.

The application level caching of data is a popular solution to reduce the database load. Data in the cache can be easily accessed and therefore improve its performance since cached data requests are not necessary in the database and can be efficiently served from the cache itself. In short, the advantages for cached data are twofold: access is extremely fast and the load in the database is reduced.

In this paper we investigate how intelligent load balancing and caching can improve performance and improve their scale in multi-stage architectures. We analyze cache statistics by logging on application server and develop a series of various application server load balancing and caching strategies. A monitoring tool has been specifically developed for the analysis of access logs and

later load balancing and caching strategies generation. The tool operates on the server, regardless of the machine. The strategy developed is mainly based on statistics of objects to be accessed in the cache. A main feature of this strategy.

The load balance and cache strategies are very closely combined to transmit a query to an app server that provides the cache data most often required for the request, that is, to increase the hit rate of the cache. The load balancing strategy contains load balancing instructions to direct customer requests. The caching strategy provides instructions for application servers on which cache items and how long they are to be cached. The focus of our strategies is also on how cache object positioning affects performance. We examine, for example, how replication of the cache affects system performance.

We also collect and develop caching and load balance strategies, which are then analyzed (i.e. if the system works) in real time. We can also configure the strategic interval of generation by means of the monitoring tool. Due to the essentials results of these strategies from analysis of the cache statistics of an application server, we offer an application-independent solution.

An existing distributed cache co-operative infrastructure increases the overall capacity of the system cache. It allows us to fetch requested data from another app server's cache in addition to the server cache where the request has arrived. We assess the performance of our strategies with our well-known E-commerce benchmark. The results have shown that efficient load balance and caching strategies can lead to significant improvements in performance.

At the end of 2005, a framework for web development was published. The Ruby on Rails framework was supported by Ruby's programming language. Ruby on Rails.-Ruby on Rails. It provided a new way for Web developers from Ruby using a 1970s architectural software model originally designed for user interfaces in desktop software applications. This model is known and forms the basis of frames such as the Model View Controller (MVC). Because of Rails ' popularity, a rush would follow in different programming languages with similar web application frameworks. The web frameworks constitute a crucial tool to meet modern demands and are working well with projects such as agile and Kanban, because of the complexity of global web pages, now known as web applications, they have been developed.

But many software companies and developers still have a little confusion about the MVC frameworks and do not know whether such frameworks for a specific project are the right choice. For example, ASP.NET and WebForms develop companies. After reading this thesis, the Author hopes that the development practices can be better understood and the adequacy for a particular development project of the MVC web development framework can be evaluated.

In its new nature, the author shall document in the entry-level framework for web development from an independent point of view the value of writing the web application, as recommended by the present institution. The project is sufficiently complex to evaluate MVC architecture libraries efficiently and offer guidance on future projects using MVCs.

The thesis gives an overview of the pattern of the design for MVC in order to assess the value and the reuse of code by using the MVC frameworks.

1.1 PURPOSE AND IMPORTANCE OF THESIS

The web's popularity and benefits have led to numerous web apps as a server platform for clients. In global environments, Web applications provide applications of all kinds. The popularity of web applications is because the app has access to all platforms from all over the world. Moreover, the centrally arranged maintenance of the web is cheap.

We rely on the following statements to ensure the fast and high - quality development of web applications:

- For most commercial applications, web applications are the first choice.
- The design and maintenance of one of the main challenges for the software industry is web applications.
- Due to the complicated nature of the system, it is difficult to maintain an application.

A MDA enhances web applications quality, speed and maintenance. Furthermore, the MDA includes steps to develop, integrate and architecturally model business models.

The JSF framework is a popular web-based MVC implementation to demonstrate an MDA-based approach to web apps. For most web applications, the MVC patterns for data separation and

presentation are used. The JSF frame provides components for server user interface for Java - based Web applications. We chose JSF mainly because it controls a ready - to - made page stream.

Models that define the user interfaces of websites and pageflow are described in Web applications. This pageflow describes all web pages of a specific page. Our approach aims to improve the clarity and understanding of a definition of a page flow process. A meta-model needs to be defined to define the structure and syntax of the models. Consequently, the meta-model defines the DSL. Every model is a meta-model instance. This meta-model is also used for model validation.

Our approach describes a MDA in the production of MVC - based web applications. We are focusing on automatic production of JSF web applications that show how the Apache Tomcat web server is working. A prototype described later is presented in this work. Additional efforts can be made to create web apps that can not only be deployed on a given web server.

Our defined meta-model enables web pages to be modeled and flowed. The controller comes with the flow-controlled JSF framework. The XML file that the controller's input is generated by our code generator. The XML is based on the given web application model. Today, the developer is responsible for the model of the MVC, i.e. access to a database. This results in manual code handling.

Our web pages, Tomcat's web server configuration file, the XML file that includes page flow information the JSF controller uses as the input, Java data storage beans and the Apache Ant XML file that facilitates deployment of the Tomcat Web Server application generated are currently generated by our generators.

WAG has several features, which can be attracted or not by developers and companies. It says that for WAG our objective is to achieve the maximum in the most small and lightest package performance, capability and flexibility. The component and the routine only works and runs on request to achieve these objectives. You call this dynamic system, and the system is very lightweight by default. Finally, when calling for the loading of code, the developer must invoke components and routines.The project is designed to comply with the MVC's loose linkage and high cohesion principles.

Some of the developers ' interesting things are below. The configuration is trivial in local development. Use local host environment to download, extract and alter certain parameters for configuration of zip file. After local development has been completed, you can easily install the app on your client server or server by simply changing some of your files to server needs and uploading files like the FTP client. Things only work due to the ability of WAG to operate a number of PHP versions.

The framework is not compliant with restrictive coding rules, which might mean that the developer does not write the best practices code but allows for the quick creation and later re-facturing of something. WAG doesn't need a terminal control line as opposed to some other frameworks, or it depends on tools like package managers or other frameworks.

The security of the framework is serious. It also joined the HackerOne project in recognizing the importance of the general security community in addition to its in the house team (CI Project Committee).

2. LITERATURE REVIEW

2.1 CHAPTER OVERVIEW

After defining the problem to be fixed in the previous chapter, this section of the report provides an overview of the J2EE platform for the online project marking system. This chapter also discusses the selection of the J2EE design and development framework.

2.2 J2EE PLATFORM

Java 2 is a Java based business development platform-independent environment for business applications. Sun Microsystems developed Enterprise Edition (j2EE). J2EE is able to support multiple Java 2 platform components to help develop web application, namely JavaBeans Enterprise, Java Servlet API, Java Server Pages (JSP) and XML Technology (J2EE (2005)).

In order to build on the new version the customer must use the same technology and the previous implementation of the project marking system with J2EE. Due to this restriction, the adequacy of alternative technologies like PHP or was not taken into consideration. The following sections focus more on issues relating specifically to the J2EE platform. NET Framework;

2.2.1 J2EE Components

The two common J2EE components for Web-based applications are Java server pages and Java server pages (JSP). Servlets are Java programs that receive and process HTTP client requests, launch server-side resources and generate answers that will be sent to a client after deployment in a serlet. The JSP-page is also compiled into a servlet which can be used for either component to achieve the same goal. JSP is primarily developed as an HTML document that includes Java code in the page. Servlets are also developed as Java classes that produce the output of HTML. These servelets have also been developed in Java.

Deptors provide increased maintenance for applications which perform complex processing–one technology may be more suitable, usually JSP, for presentiment-oriented applications depending on the type of application being developed. In some cases the application fitting into both

categories, such as the Online Project Marking System, requires a complex interface with a significant amount of corporate logic. In these cases, a more efficient solution can be achieved by combining the various J2EE technology.

2.3 SOFTWARE DESIGN PATTERNS

Software designs are predefined solutions to recurrent problems in software engineering. A range of designs have also appeared at the Sun Java Center for J2IE Development Projects [Alur et al (2003)], while many of these patterns are available on various platforms. As Calder and Winn (2002) pointed out, respect for software design patterns can strongly support development through the provision of expertly tested solutions and accepted standards of good practice.

2.3.1 The Model View Controller (MVC) Pattern

The MVC model is a design model used in a variety of software development projects and platforms. The MVC pattern is a design pattern. The principal concept behind MVC is that the application design problems are divided into three layers–model, view, and contoller–. The model component administers the business logic and application status, provides the user with the display model and interprets interactions with the image through modeling action. Figure 2.1 shows the connection of the MVC layers.

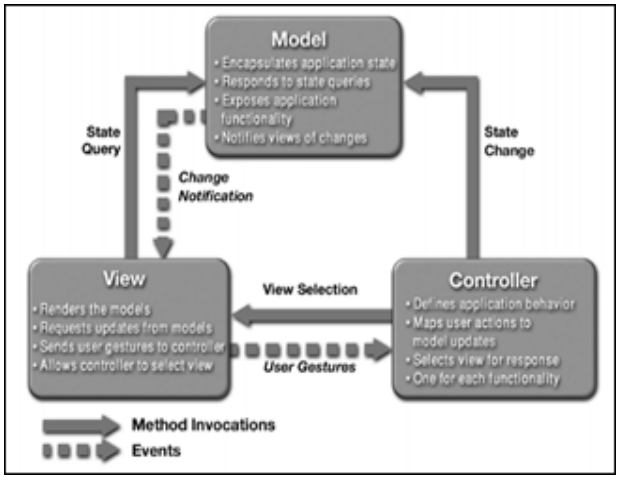

Figure 2.1: MVC Layer Interaction

Dividing design concerns using MVC exposes a variety of options, including the ability to build various alternative views using the same model and controller. In addition, collaboration can be approached with greater confidence and facility when different elements are clearly separated.

J2EE technologies can also be combined to improve the efficiency of application designs. For J2EE, the so-called Model 2 incorporates the MVC Design, which contains Javanese beans representing the model layer, JSP pages for view and Java servlets for the application manager. A controller serlet uses one or more JavaBeans when a Model 2 application is sent to initiate changes in the model's state. In the event of a reply, the controller servlet selects a suitable JSP page for viewing. Figure 2 presents the diagram of a typical application of Model 2.

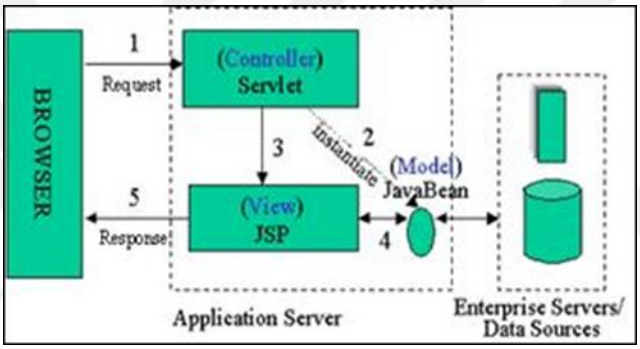

Figure 2.2: A Model 2 Application

Although the typical Model 2 frame can effectively disconnect MVC design problems, the controller component can be criticized for a lack of maintenance. The model can be reused and the components viewed for different operations using a servlet collection to offer to the application controller. However, this has the disadvantage of replicating code in all servlets for common control operations. In more important developments, the effect of replication is lower maintenance, in which further application components are affected when an amendment is required.

Please give an example of a request containing 30 checked servlets, fifteen of which are shared–page 1.jsp. The application takes advantage of its opportunity to reuse the single view component by sharing it with different operations. However, changes to the application structure would lead to maintenance of all fifteen control serlets, for example by transferring the five operations to another view page2.jsp.

Offutt (2002) identifies maintenance as one of the most important indicators for high quality web applications. The front controller pattern [Front Controller (2005)] J2EE is the main feature of the typical Model2 application.

2.3.2 The J2EE Front Controller Pattern

The Front Controller approach centralizes the handling of requests in the J2EE application. Centralization is achieved by providing a single entry point for any requests, the Front Controller Servlet. The front controller usually uses an Action and View Management dispatcher. The dispatcher is also known as the application controller. Action management refers to the method of finding actions to respond to specific requirements, whereas view management concerns the selection and transmission of suitable points of view to respond to requests.

Figure 3 shows all customers' requests and delegates to the distributor for managing action and viewing by the Front Controller.

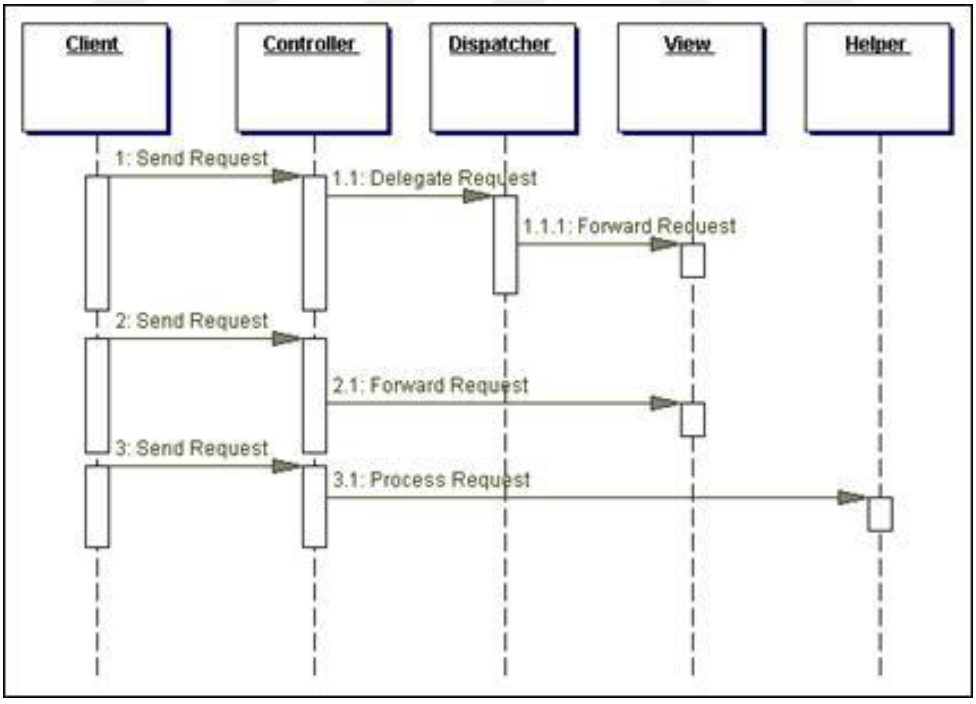

Figure 2.3: Front Controller Sequence Diagram

A main advantage is the centralized controller's ability to utilize common control logic for all requests. As we have said, management of actions and visuals is central, but many other

operations such as authentication, error handling, logging and persistence layer connections are available. The centralization of these operations mainly reduces the duplication of the control logic. Look back at the typically Model 2 approach, as a result of the MVC design pattern, the Front Controller provides equal and higher maintenance levels by minimizing duplicate code.

2.3.3 The Business Delegate Pattern

The pattern of the business delegate is also an example of MVC's separation approach. The Business Delegate Pettern offers a means of untieing business components of the application logic (controller) that it uses. Separation is achieved by providing the Business Delegate intermediate class, which offers several Interface methods to ignore the underlying complexities of business services interactions. The programming interface for the application.

Figure 4 shows how the representative of the company functions as an intermediary between the customers and the business sector. Interactions with the Business Delegate's customer interface trigger business services and appeals.

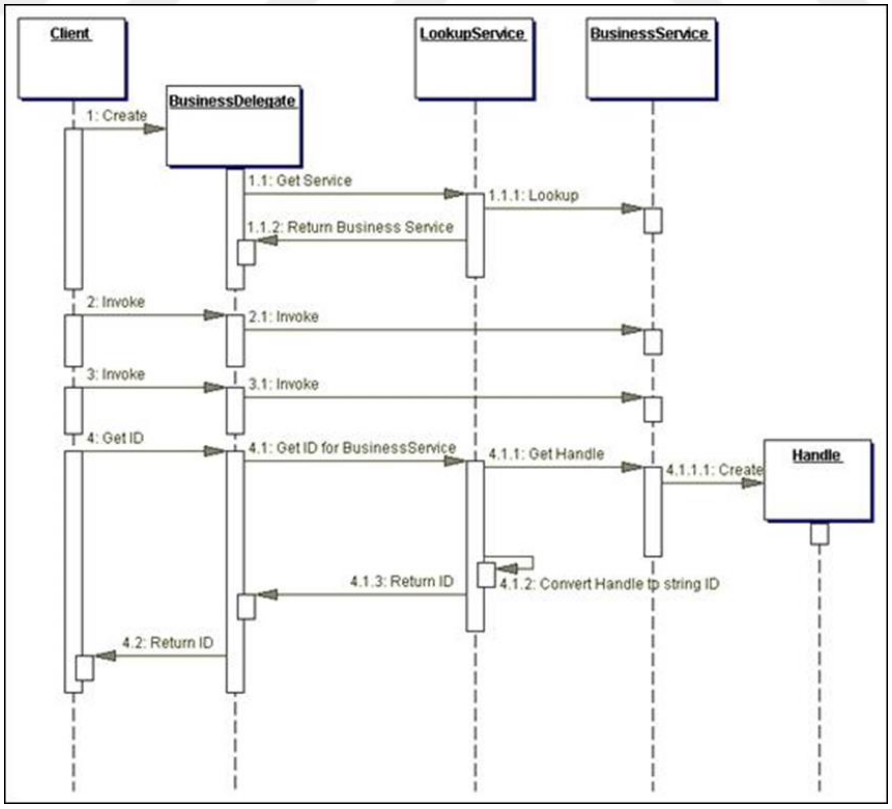

Figure 2.4: Business Delegate Sequence Diagram

We have a number of advantages by adopting the business delegate pattern. The first step is to improve application maintenance. A single maintenance point is the central implementation of the corporate level in the business representative. In addition, the clear separation between business and application logic reduces the potential to change the level of business affecting other components of the application. The Business Delegate also provides a standard interface to enable customers to interact with company services, ensuring that business services are used consistently in the application.

2.4 J2EE DEVELOPMENT FRAMEWORKS

We have seen how patterns can be essential if good design practices are to be implemented. Software framework frameworks can also offer a range of benefits, such as the ability to reuse standard application components and, in some situations, to provide one or more levels of the application.

Johnson (2005) stresses J2EE's failure to develop appropriate framework for company applications and the lack of support leading certain (typically larger) companies to develop their own home-grown framework. However, the increasing request for standardization in this field has led to a number of frameworks in open source.

2.4.1 Open Source J2EE Development Frameworks

Johnson (2005) describes some of the advantages of a well - designed open source framework.

1. It first allows developers, by providing reusesible custom components, to create applications with a minimum of code.

2. The use of a framework also allows for consistent and clearly defined applications to be helpful for collaborative developments.

3. In addition, Open Source Frameworks are advantages for more detailed testing and distribution in comparison with in - house solutions, including detailed documentation and

example applications. In addition, it implies higher quality because they are used more frequently.

At the time Jakarta Struts was written in terms of web - app development, the most widely used open source framework. This framework is discussed in greater detail in the following section.

Another example of the open-source framework that allows access to the continued stored Java apps in the business layer is Hibernate developed by JBoss, Inc. Hibernate deals with questions and complexities in accessing the JDBC database by providing Java objects with associated database table mapping.

Whether or not the number of open source architectures which will support J2EE development is still growing and whether Sun Microsystems is addressing Java's apparent weaknesses in its framework for development remains to be seen.

2.4.2 Jakarta Struts

In the previous section, the concept of an open source development framework for J2EE has been discussed. Jakarta Struts is an example of a framework used in the development of the J2EE Web Application. Struts is classified by Husted et al (2003) as the' invisible foundation collection' which helps developers to make a coherent application in raw materials such as databases and websites.

The Java web application controller component is supplied by Struts, the project of the Apache Software Foundation, and supports integration with a number of other technologies to enable model development and view component development. In the above Hibernate framework for the model component, Struts incorporates JD BC, EJB and third-party technologies. In order to support the development of a view component, Struts provides five tag books, which are classified as view supports and integrated with JSP (JavaServer Standard Tag Library) and JSF (JavaServer Faces).

The Struts controller consists of 5 core classes of components. It's these classes programmed and configured by developers to achieve the necessary features when creating the Struts - based application.

1. ActionForward is used to define hyperlinks in a Struts application. In the XML configuration file for each ActionForward object, the logical name and path are generally defined. ActionForward is greatly beneficial because its logical name is used in the application to refer to other components, such as the JSP page. That means that changes to the hyperlink may be caused throughout a Struts application by changing ActionForward from a configuration file.

2. The ActionForm class handles HTML form input. In the ActionForm, rules can be programmed or set via XML can be validated. In the event of an input error, the ActionForm displays the form again with the right error notifications. ActionForm is a Java Bean with features reflecting the fields of HTML input. Struts automatically adds HTML form data to the associated properties and thus prevents developers from writing code for the purpose of extracting formal data from the HTTP request. The ActionForm is a huge benefit.

3. Action class is defined as data processing in HTML form. When the ActionForm has been verified, struts passes an ActionForm to the action given in the HTML action parameter for processing. The action class is tailored with the ActionForm data to the specific features of developers. An action ends when an ActionForward item determines the user's response is returned to the controller.

4. The ActionMapping class offers a means to associate Action objects in an HTTP request with a URI. An Action Mapper indicates the path of the application and the execution action when the request is received. The ActionMapping can be used to combine the action with ActionForm and a collection of ActionForwards. ActionMappings has the main advantage of being able to reuse multiple ActionMappings actions.

5. Struts delivers the Front Controller model outlined in Section 2.3.2 by providing its ActionServlet class. The ActionServlet is the only entry point for all web application requests that are deployed in a service container. The ActionServLt's main function is to parse ActionMappings located in the controlsettings file for the purpose of finding and instantaneously moving the action and action associated with the URI query. The ActionServlet also provides ro

Les-based selection and security

Table 2.1: Core Struts Classes As They Relate To MVC

Class	**Description**
ActionForward	A user gesture or view selection
ActionForm	The data for a state change
ActionMapping	The state change event
ActionServlet	The part of the Controller that receives user gestures and state changes and issues view selections
Action classes	The part of the Controller that interacts with the model to execute a state change or query and advises the *ActionServlet* of the next view to select

Figure 5 shows how customer requests are handled after the core classes of struts have been defined.

The Struts framework has many advantages, the first of which is considerable flexibility in terms of integration into model and vision components, as Husted et al (2003) explained. Furthermore, the framework is comparatively lightweight to developers with only five tag libraries and five main classes. Another important factor is support from suppliers and developers, documentation and example applications. His high level of support is another important factor. Since the frame is open source, suppliers can not be guaranteed support and developers ' communities can not necessarily supply the right information. However, this should be taken into account.

Some have criticized Struts for implementing its MVC design pattern. Knight and Naci (2002) point to the potential for blurred application control and logic in the Action classes, which is in violation of MVC principles. However, any developer wishing to apply the MVC principles is certain to avoid such bad practices. It is not possible.

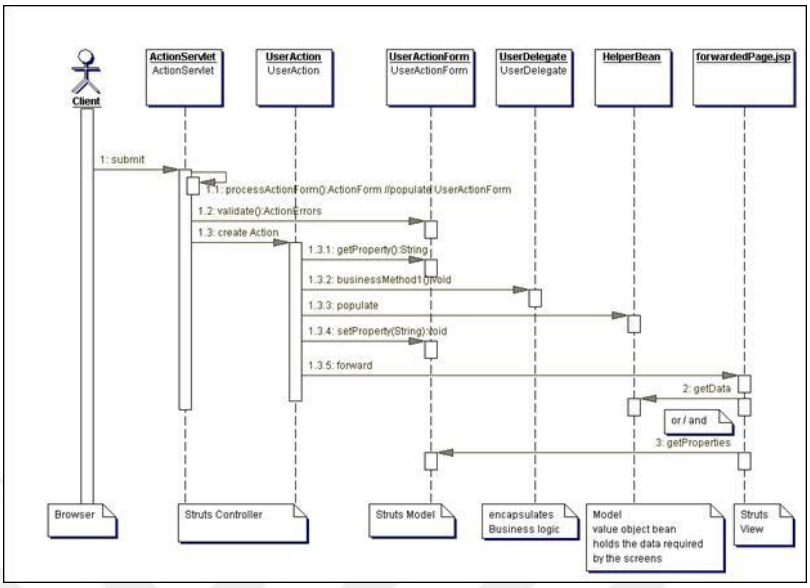

Figure 2.5: Struts Sequence Diagram

1. A HTML form is submitted by the client to the Struts request and the ActionServlet captures the HTTP request. The ActionServlet finds the ActionMaping linked to the request URI, adding HTTP request form data to the specific ActionForm. The ActionServlet calls the ActionForm validation method, which does not return errors, to confirm validity. The ActionServlet moves the ActionMapping action and transmits the populated ActionForm. The Action processes the form data involving business layer interactions, and concludes with a Action Forward that returns the view -"forwadedPage.jsp."

2. The view deletes and displays data held by HelperBean's Action.

3. In the view, the data in the action Form is extracted and shown.

The Open Symphony WebWork project to explore alternative J2EE web application frames is the current potential competitor in Struts. WebWork is built on the Open Symphony XWork framework and claims to have a number of fields in its documentation that are superior to struts. For example, Struts installs one Struts action in respect of all requests to ensure thread-safe operations of Struts and all its resources. WebWork claims that the mechanism is restrictive, and that instead, a new WebWork action is introduced for every request that removes the thread security requirement.

Despite the argument of WebWork, the Struts are widely documented on the Internet and printed literature, a sign of its popularity as a framework for website development within J2EE. It does not mean that WebWork can become a stronger competitor in the future as it develops.

3. METHODOLOGY

Software that generates software for the Web Application Generator (WAG) from problem descriptions rather than traditional programming. It can be used more easily and at a higher level than a language like C / C++ or Java that has been highly developed. A statement or description may produce an enormous routine or a complete programme. Complex program development generators allow the expression of if-then-else programming together with simpler descriptions of inputs and necessary outputs.

The problem with these high-level systems is that the resulting code is either too bloated or too slow, or some functions simply can not be carried out at all. Consequently, commercial programs in these languages are seldom written. For enterprise information systems, application generators are often only used for the development of prototypes that are reprogrammed in a later language, like C / C++ or Java.

Computers can absorb excess code from higher-level products as they run faster. Although it hasn't been done yet, one day most developing tools at a very high level will be employed, which relegate a small number of highly qualified people to the traditional programming language. CASE, application and foundation classes. See CASE.

3.1 PATTERN THE MODEL SIDE (MVC)

Web applications contain a variety of user information content on a variety of pages. The development teams are responsible for the design, implementation and maintenance of such web apps. This is why multiple types of user interface, for example, are supported. Pages to users of HTML and Java Web developers. This is a problem requiring different points of view from the same data. Additionally, the same data may be updated with different user interfaces. The elements that provide the web application's core functionality should not impact the support of multiple views and interactions.

To correct these things, the MVC pattern is used by distinguishing between core business function and the presentation logic. This division allows you to see the same data multiple times. This allows several customers to be implemented, tested and maintained.

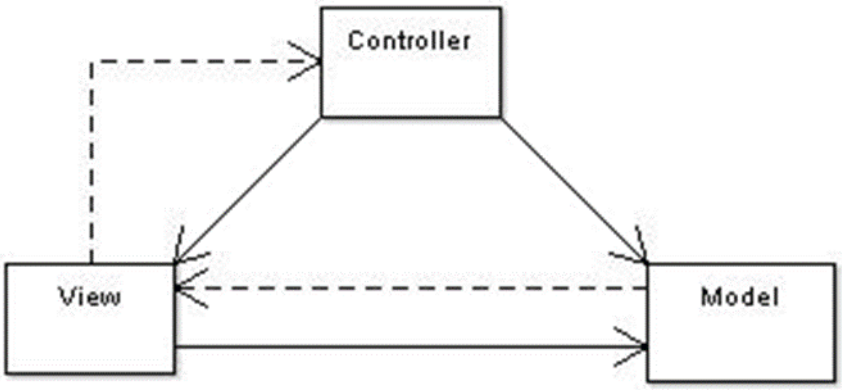

Figure 3.1 . Demonstration Of The MVC Pattern

- **Model**

The model encapsulates an application's functional heart. It displays the data and allows data to be accessed.

- **View**

The display takes charge of rendering the models data, usually elements of the user interface. The presentation of the data is specified precisely. If the data in the model has changed, the view must update the data presentation. Usually the view reads, as the view should not be changed only access to the model.

- **Controller**

It is up to the controller to call the model methods for data modification. There is an equal field between the controller and view. The controller cannot copy data values from the model to the display, but can insert values into the model and show that the model data has been changed. A new view that the user must display is also available for the controller, like a website that shows the results.

The View is the actual MVC pattern based HTML document for web applications, and it's the controller who controls the content and page flow within the HTML code. The model consists in a database or XML file containing current content.

The following scenario shows the actions when the user interacts:

1. UI.
2. The opinion acknowledges that actions have been taken and calls a registered calling method for action
3. The View refers to the appropriate Controller method
4. The controller is modeled.
5. The Model informs all views of the change in data. The Controller can also update the view in Java technology - based applications.

The benefits of MVC are:

- Substitutable user interfaces

The alternative user interfaces for the same model can be replaced by different views and controllers.

- Multiple views of the same model simultaneously
- Views synchronized
- Easier to modify user interface
- Facilities to test

The disadvantages of MVC are on the other hand:

- More complexity
- Close connection between View and Model controller:

3.2 THE GENERAL WAG PARTS EXPLORING

Although MVC is an integral component of WAG, its architectural design model is not only a Web development framework. It is composed of other important elements (see figure 7) which act on data sent throughout the system.

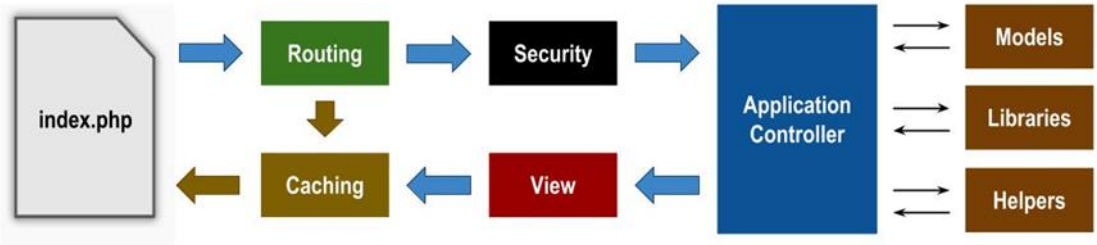

Figure 3.2: A Typical Application of Different Classes and Modules

3.2.1 URLs and Routing

Standard query string parameters in traditional web applications are normally used to 'Post' and 'Get' method on data. Examples include:

http://www.example.com/Blog/Posts.php?Year=2015&Month=10&Day=11

The segment-based approach is mainly used for WAG and other MVC frameworks. The segments (including any directory path) are also called Uniform Resource Identifier (URI) segments. They are the class, the method, and all the class arguments. For example:

http://www.example.com/blog/posts/2015/10/11

The above URL contains ' blog ' for the controller, ' post ' and ' argumentation ' for the method is ' year / month / day. The segment approach allows URLs to be humane and searchable.

When a framework, like WAG, routes a URL, the default action with the URI segment is modified or rewritten. In the application / config / routes.php configuration file, routing can be configured. For example:

```
$route['product/(:num)'] = 'catalog/product_lookup_by_id/$1';
```

Instead of the catalog controller class and descriptive method, the segment can be replaced by the ' Product ' and its number here. However it will verboze a ' Product lookup by I d ' represented by the variable ' $ 1. '

3.2.2 Caching

The caching of web pages is a storage method on a part of a page, which is normally not modified when refreshing, so that an app is reloaded faster. In WAG, page by page caching is carried out. The optional caching is not default and it takes several minutes for the controller to use it as the expiry argument. Cached files are usually placed in an application / cache/ folder that requires writable authorisation for that web server directory. It is claimed that files accessed from this cache folder achieve performance close to the static web pages.

3.2.3 Security

In order to use ' best practices ' in Web security, WAG has various internal security features. For instance, the type of characters that malicious data are allowed in URI segments is restricted.

In the initialization process, global variables are also discovered by $GET (see PHP Globals Registry) $POST, $REQUEST and $COOKIE. This prevents mixing variables with other locally defined variables in a different HTTP request method or, worse, $GET requests, for example, which attempt to manipulate code to circumvent various logic controllers, like the authentication. The security in PHP is the same as the global register, but in some previous server configurations the settings are "enabled."

Error messages in development are essential, but once they are received and are available to the public, developers will be advised to deactivate this feature otherwise they risk accessing

sensitive information from the end user. In place of using the php.ini on a server, WAG can easily configure it in the application. In the index.php folder various operating modes are developed, tested and produced, and the ENVIRONMENT constant as the root folder is defined.

WAG provides a means to filter possible JavaScript (XSS) malicious output to hijack user browser sessions. There is therefore a need to cleanse every result that third parties have received (for example, user comments or even possible infected photo uploads). For example:

```
$data = $this->input->post('comment');

$data = $this->security->xss_clean($data);
```

The cross-site request forgery (CSRF) is another type of attack that can affect users. If the form helper form open) (procedure is being used, WAG inserts a hidden CSRF field automatically into the web forms. This thwarts CSRF attacks with a once-in - one key token that an attacker can hardly guess when trying to duplicate the action of the form.

The WAG Validation Library provides a more precise and easier way to validate and develop data rather than calling for several individual validation or preparation functions with PHP in particular. The validation can be defined in the controller or model; the controller is generally called. (Testing the data according to type, length, presence, similarity, etc) (the following is an example of the rules). The rules are defined according to various methods.

```
$this->form_validation-
>set_rules('username','Username','trim|required|min_length[5]|ma
x_length[12]|is_unique[users.username]' );
```

(There is an example of the last check rule that even contains requirements for an automatic database column to check that the data in the username column is unique. When the validation errors are placed into the view, error messages can be received automatically (the method passes three argument): the name of the forms columns, the label of each rule separated by the tubes character. Misprints may be adjusted as well. The method is used after returning the error to repopular a form with the original input.

A number of malicious injection solutions are provided by WAG Database. It offers also a simplified query binding process that allows the system to assemble and escape the developer's database query by separating database syntax and data during state preparedness. The process is also called prepared data. For example:

```
$sql = "SELECT * FROM your_table WHERE id = ? AND status = ?"; $this->db->query($sql, array(7, 'available'));
```

Then, it has an Active Record - like query builder, which provides an easy interface for automated database operations and work on various data base engines. For example:

```
$query=$this->db-
>get_where('positions',array('project_id'=>$project
_id,'day'=>$day));
```

Does the MySQL declaration equivale:

```
SELECT * FROM positions WHERE project_id = $project_id AND day = $day;
```

3.2.4 Views

Views is an essential component of the MVC design. Apart from the layout are applications (business logic, logic on data accesses and validation logic). Only web pages (header, footer, menu) or fragments of a webpage can be seen as visuals. In contrast to traditional web applications, however, the view is not directly called by the end user. Checks are always loaded to make the presentation types more flexible, as instructed by the developer. As mentioned above, WAG processes multiple calls intelligently in order to charge each fragment of a view in the order in which it is named. An associative array or object is used to define and transmit variable information from the controller. The key values are used as variables and/or arrays directly (as transmitted by an associative array). WAG views support your own template syntax to prettify a regular HTML template syntax, PHP. The WAG template syntax, however, needs a parsing process to slow down and naturally it takes time for the developer to learn too.

3.2.5 Models

Models like views are in the architectural model of the MVC. A model is basically a class that works in a database with data. It is not obligatory to use it because of the flexibility of the WAG framework. However, developers use the traditional MVC approach generally. If there is a model it is called as follows in the controller class:

```
$this->load->model('my_model_name');
```

You can then use your class name to access:

```
$data['query'] = $this->my_model_name->fetch_database_info();
```

Even if the model usually works with the data access logic, the validation rules are generally defined and called by the controller. The following chapter shows an example of this.

3.2.6 Controllers

The controller is a class, and is a part of the architectural design model just like the model and viewing classes. The controller shall serve as data intermediary for processing HTTP requests between view and model classes. When a user interacts with an application, you can also access the controller to be considered the application access point of the controllers. The URI segment and the routing sub-chapter were explained in order to expand the concept of an access point. A URI segment (with optional arguments) is only one part of the URL which can call a defined method. Generally, this class method loads several libraries and helpers, arranges model calls and loads the view for the user during data transfer.

3.2.7 Libraries

Library classes (in the case of WAG) extend the functionality of the framework. As already mentioned, not just an architectural design is a comprehensive web application frame, but its libraries as well. They are a toolkit or a add-on and can contribute to the rapid development of a web application. The library of WAG lies in the system / bibliothèque directory and the

application / bibliotheques are located in user-defined libraries. In the controller, libraries are initialized, such as:

```
$this->load->library('library_class_name');
```

Once the developer has been loaded, his processes can be documented.

3.2.8 Helpers

Helpers help a developer with various tasks to develop a Web application. They are similar to the library, but instead of classes similar to PHP's built - in functions, they are simple and procedural. This is why the helpers perform certain tasks and have no dependencies.

No wonder that the developer has to load or even have an overview of the required supports within a controller. However, for example, this is quite trivial:

```
$this->load->helper('url');
```

As any other PHP function after this, the helper can be used:

```
echo current_url();
```

More than 20 assistants are in the context. CAPTCHA helpers, date assistants, url helpers, form helpers, email assistants, among others. Auxiliaries that provide a frame will be saved in the system / helper directory. You can also define the support framework for your own assistants or expand it. In this case, customer helpers or extended helpers are saved.

4. IMPLEMENTATION

4.1 DESIGN AND ANALYSIS

A) Problem Solution Analysis

The approach from the bottom up can achieve a generic pattern. The generic pattern is achieved through a number of platforms to analysis the source code. Before a generic pattern and pattern was developed for each platform and technology, the software code for each platform and associated technologies was analyzed.

Two stages are the main structure of a dynamic web application:

1) The dynamic development stage and 2) the production stage of codes. For the development of dynamic web pattern, also, two main sub - stages for the analysis and design of dynamic web development patterns.

We assessed the MVC architecture model concept model and its specification for each layer during the analysis stage in order to have the dynamic pattern web application well developed. We examined, for each platform and technology, the basic design patterns and specified patterns during the design phase.

The pattern in generic patterns defined, which can interact with specified rules requiring a specific technique. Furthermore, define standard patterns for all technology should be applicable. By its purpose, the standard pattern is defined as standard generic patterns of communication with different technologies. The generic pattern for development is used for generating program code during the implementation phase.

B) Dynamic Web Results Basic Pattern Generic Analysis

The results of the analysis leading to a generic pattern of dynamic websites are different technology platforms:

1. The developed pattern should contain a high abstraction level in each layer of the MVC and can also be applied in the program code and interact with another layer in each layer.
2. The details of each technological application interact with a different layer of different techniques. The technology specimens cause very exact interactions and could ends to the lack of addition of MVC layers. Interaction between layers through interface or mapping is the solution for integrating the MVC layers. The communication between MVC layers through txt file mapping is designed in this study.
3. Basic pattern definition that applies to the technology of all developers. The basic design is defined as standard generic patterns of interaction with different technologies.

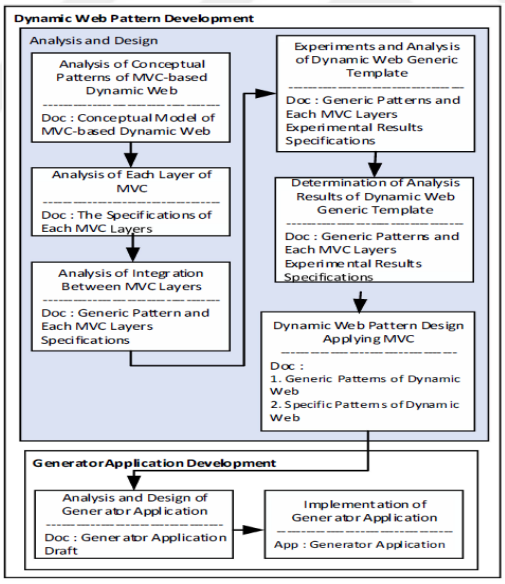

Figure 4.1: Analysis Phases

4.2 THE IMPLEMENTATION PROCEDURE

The applications analysis involve the structure of each MVC layer, libraries, platform files and the implementation of technology, and the characteristics and interaction of each project. Requirements Analyzer for the application. MVC architecture is an application with the ability to produce a web code dynamic on different platforms (Java EE, .NET, PHP). The web

application generator is known as the web application generator. This program is called "Web Application Generator." Dynamic web functions like reading, creating, updating or removing can be provided by the WAG application. WAG can produce a database program code. A web program with the relevant technical podium and user selection is a WAG application output.

Figure 3 process flow description is as follows: 1) the database is ready, and then the data bases are loaded by WAG. 2) WAG can find all the data identities associated with the database from the database already loaded, like table names, tables' relationships, primary key for each table, attribute names, attribute types, and more. 3) Program code that is generated by WAG according to the platform technique and users required, once all the relevant identity databases have been obtained. WAG produces the code of the individual MVC layers and generate all settings (including database settings, application configuration, layers integration configuration, etc.), library loading, application structuring, etc. The code generation process is implemented by WAG. WAG uses package configuration files for application support, for example library and templates during the application code generating process, where the files are bundled when the applicable WAG is installed.

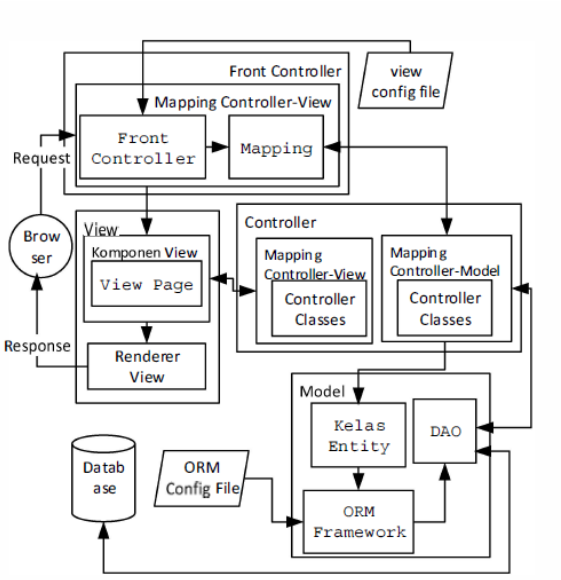

Figure 4.2: Generic Pattern of Dynamic Web Architecture

1. **Database Loading Algorithm**

   The flow of processes described in Figure 4 as follows: (1) connect to the database; (2) access the databank as connected to that database for the identity related to that particular database, such as attribute names, key of each table, attribute type of data, names, tables-to-person relationship; During program code generation, the process of mapping can be done by itself.

2. **Code Generator Program Algorithms**

   The process of application code generating points to the data generated in the list algorithm of Figure 4. Dynamic web applications matching selected technology users and platforms are generated from List data. WAG generates the code in application generation on each MVC layer, creates all settings (e.g. database configuration, application settings and layer-to-layer settings), loads and structures application libraries. WAG depends on software files for supporting such as package and libraries which are templates when the program code production process takes place. The application generation output is a web application that is already functional and which creates, reads, updates and deletes the application.

3. The WAG user interface.

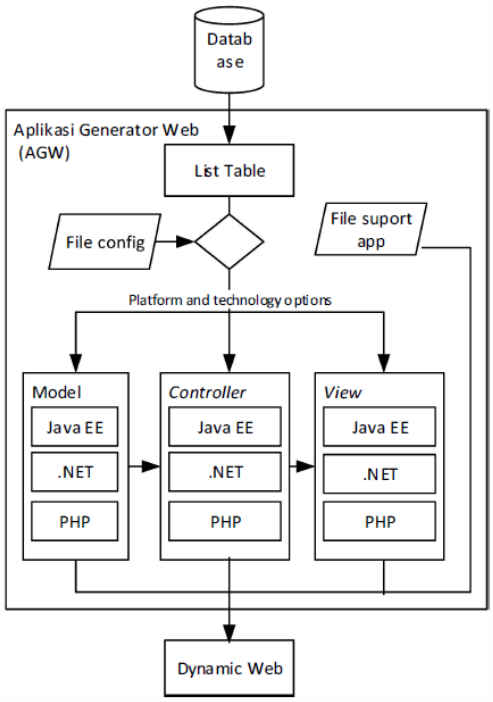

Figure 4.3: WAG Architecture

4.3 TESTING

Using case studies, implementation of WAG was tested. The tests are aimed at validating the generic pattern that a dynamic web has developed. A functionality and conformance (conformance) test were used to validate generic patterns. Tests were conducted to confirm the dynamic web applications created by WAG's ability to properly create, read, update and delete with the features in the set. The application code that was generated for each platform was by functionality testing. The code was generated on each platform by experimental combinations of all techniques. The appropriateness tests in this study are to test the adequacy among generic patterns by producing dynamic web code.

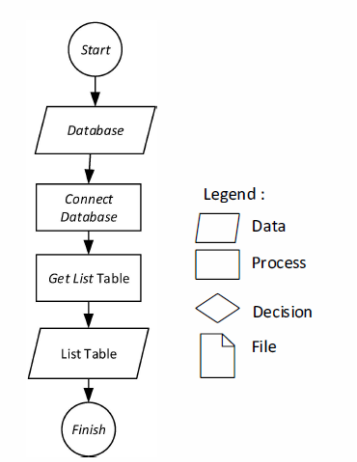

Figure 4.4: Flowchart of Algorithm of Database Load

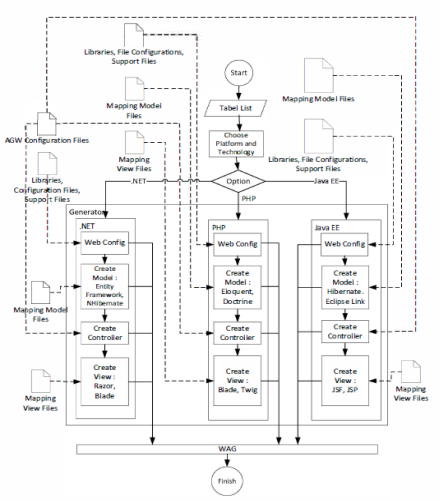

Figure 4.5: Algorithm o/Code Generator Program

5. CONCLUSION

A dynamic web pattern with multiplatform has been proposed in this study, based on this proposal and a program generator. We hope that our research will improve productivity of the developers. Different platforms and certain technology were examined for each platform. We proposed a basic pattern for recognizing variances in these techniques integrated at different levels of MVC, in keeping with our initial study. Mapping is required to integrate every MVC layer. The map shows the interaction mechanism between the layers of MVC. We think the addition of new technologies can contribute to the programming of our dynamic generic web pattern.

5.1 SUGGESTIONS

Our approach encompasses only a small part of web app modelling and generation science. For that reason, in the future there is still a lot to be done. This section describes how through further work we can expand and improve our approach.

5.1.1 Meta-Model

Today, our defined meta-model offers web pages and web application pageflow facility. The model can be extended by providing web pages that enable the developer to define and locate web pages that include components of the user interface, such as areas. TOP, LEFT, CENTER, etc. The MVC model, which collects necessary data, cannot currently be modelled by the developer. This is also a major workplace. We need to assess if the separation of the meta-model in several metacodels is meaningful when offering the facilities for modeling the MVC pattern. Related works, as described in the Related Works section of Section 3, offer several meta - models responsible for View modelling, Controller modeling and MVC mode modeling.

5.1.2 Model Transformations

At present, only a JSF web application is transformed into a modeled Web application. Apache Tomcat Web Server uses the generated JSF Web application. We will be creating more model transformations in future, not just for the web server Apache Tomcat, so the modeled web application should be developed. We must address the question of how to provide the MVC pattern controller to generate web applications besides JSF. At this point, the Java Servlet FacesServlet from the JSF framework controls the web application pageflow. For this Java JSF server, we provide a template for the input to the XML file. In addition, we wish to provide the production of non-Java-based Web applications, for example. Web apps for PHP.

REFERENCES

[1] I. Sarker and K. Apu, "MVC Architecture Driven Design and Implementation of Java Framework for Developing Desktop Application," International Journal of Hybrid Information Technology, pp. Vol.7, No.5, 317-322, 2014.

[2] 1. Chhikara, "A Web Architectural Study of HTML5 with MVC Framework," International Journal of Advanced Research in Computer Science and Software engineering, pp. 451 - 454, 2013.

[3] F. K. Frantz, "A Taxonomy of Model Abstraction Techniques," in Simulation Conference Proceedings, 1995. Winter, Arlington, 1995.

[4] V. Tran, 1. Vanderdonckt, M. Kolp and S. Faulkner, "Generating User Interface from Task, User and Domain Models," in Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services, 2009. CENTRIC '09. Second International Conference on , Porto, 2009.

[5] U. Zdun, "Dynamically Generating Web Application Fragments from Page Templates," in SAC '02 - Proceedings of the 2002 ACM Symposium on Applied Computing, Madrid, 2002.

[6] A. Loh and M. Robey, "Generating Web Applications from Use Case Scenarios," in ASWEC '04 Proceedings of the 2004 Australian Software Engineering Conference, Australia, 2002.

[7] H. M. Hao and A. Jaafar, "Tracing User Interface Design Pre-requirement to Generate Interface Design Specification," in Electrical Engineering and Informatics, 2009. ICEEI '09. International Conference on (Volume: 01), Selangor, 2009.

[8] A. Ginige and S. Murugesan, "Web engineering: An Introduction," IEEE Multimedia (Volume: 8, Issue: 1), pp. 14 - 18, 200l.

[9] British Columbia Institute of Technology. 2016a. CodeIgniter URLs. Accessed 5 18, 2016.

[10] Daigle, L. 2016. 30 Years of TCP -- and IP on everything! Accessed 5 18, 2016.

[11] Davis, I. 2008. What Are The Benefits of MVC? Accessed 5 18, 2016.

[12] Borodescu, C. 2013. Web Sites vs. Web Apps: What the experts think. Accessed 5 18, 2016.

[13] Berkeley, UC. 2004. Model-View-Controller: A Design Pattern for Software. 06.

[14] Apache Software Foundation, The Jakarta Site - Apache Tomcat, HTML, 2007, http://tomcat.apache.org.

[15] Thomas Stahl and Markus Voelter, Modelgetriebene Softwareentwick-lung: Techniken, Engineering, Management, ISBN 3-89864-310-7, dpunkt.verlag GmbH (2005).

[16] Jennifer Ball, Debbie Carson, Ian Evans, Scott Fordin, Kim Haase and Eric Jendrock, The JavaTMEE 5 Tutorial, For Sun Java System Appli-cation Server Platform Edition 9, Sun Microsystems Inc, Santa Clara, California 2006.

[17] Robert Eckstein, Java SE Application Design With MVC, http://java.sun.com/developer/technicalArticles/javase/mvc, Sun Microsystems Inc., 2007.

[18] Joaquin Miller, Jishnu Mukerji, MDA Guide Version 1.0.1, Object Man-agement Group (OMG), 2003.

[19] Joaquin Miller, Jishnu Mukerji, MDA Guide Version 1.0.1, Object Man-agement Group (OMG), 2003.

[20] Object Management Group (OMG), Meta Object Facility (MOF) Spec-ification, April 2002.

[21] Object Management Group (OMG), Unified Modeling Language, http://www.uml.org/.

[22] H.-W. Gellersen, M. Gaedke, Object-Oriented Web Application Develop-ment, IEEE Internet Computing, 3(1), Jan.-Feb.1999.

[23] Jonatan Alava, Tariq M. King, and Peter J. Clarke, Automatic Valida-tion of Java Page Flows Using Model-Based Coverage Criteria, Proceed-ings of the 30th Annual International Computer Software and Applica-tions Conference (COMPSAC 06), 2006.

[24] Stefano Ceri and Piero Fraternali and Aldo Bongio, Web Modeling Lan-guage (WebML): a modeling language for designing Web sites, Diparti-mento di Elettronica e Informazione, Politecnico di Milano, Italy, 2000.

[25] Pierre-Alain Muller and Philippe Studer and Jean Bézivin, Platform Independent Web Application Modeling, Lecture Notes in Computer Sci-ence, 2003, ISSU 2863, pp. 220-233.

[26] Sam Chung, Yun-Sik Lee, Modeling Web Applications Using Java And XML Related Technologies, Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS 03), 2002.

[27] S. Meliá, C. Cachero, An MDA Approach for the Development of Web Applications, In Proc. of 4th ICWE 04, LNCS 3140, July 2004, 300-305.

[28] Santiago Melia and Andreas Krau and Nora Koch, MDA Transforma-tions Applied to Web Application Development, Proc. 5th Int. Conf. Web Engineering (ICWE'05), volume 3579 of Lect. Notes Comp. Sci., pages 465-471, 2005.

[29] Eclipse.org, Eclipse Modeling Framework, http://www.eclipse.org/emf.

APPENDIX

Home/Index.cshtml

```
<section class="content-header">
   <div class="header-icon">
      <i class="fa fa-tachometer"></i>
   </div>
   <div class="header-title">
      <h1> Dashboard</h1>
      <small> Dashboard features</small>
      <ul class="link hidden-xs">
         <li><a href="@Url.Action("Index","Home")"><i class="fa fa-home"></i>Home</a></li>
         <li><a href="@Url.Action("Index","Home")">Dashboard</a></li>
      </ul>
   </div>
</section>
<!-- page section -->
<div class="container-fluid">
   <div class="row">
      <div class="col-lg-3 col-md-3 col-sm-6 col-xs-12">
         <div class="panel cardbox bg-primary">
            <div class="panel-body card-item panel-refresh">
               <a class="refresh" href="#">
                  <span class="fa fa-refresh"></span>
               </a>
```

```
            <div class="refresh-container"><i class="refresh-spinner fa fa-spinner fa-spin fa-5x"></i></div>
            <div class="timer" data-to="@ViewBag.StaffCount" data-speed="1500">0</div>
            <div class="cardbox-icon">
              <i class="material-icons">supervisor_account</i>
            </div>
            <div class="card-details">
              <h4>Our Staff</h4>
              <span>10% Higher than last week</span>
            </div>
          </div>
        </div>
        <div class="col-lg-3 col-md-3 col-sm-6 col-xs-12">
          <div class="panel cardbox bg-success">
            <div class="panel-body card-item panel-refresh">
              <a class="refresh" href="#">
                <span class="fa fa-refresh"></span>
              </a>
              <div class="refresh-container"><i class="refresh-spinner fa fa-spinner fa-spin fa-5x"></i></div>
              <div class="timer" data-to="@ViewBag.StudetnsCount" data-speed="1500">0</div>
              <div class="cardbox-icon">
                <i class="material-icons">directions_run</i>
              </div>
              <div class="card-details">
                <h4>Students</h4>
```

```
                <span>10% Higher than last week</span>
            </div>
        </div>
    </div>
    <div class="col-lg-3 col-md-3 col-sm-6 col-xs-12">
        <div class="panel cardbox bg-warning">
            <div class="panel-body card-item panel-refresh">
                <a class="refresh2" href="#">
                    <span class="fa fa-refresh"></span>
                </a>
                <div class="refresh-container"><i class="refresh-spinner fa fa-spinner fa-spin fa-5x"></i></div>
                <div class="timer" data-to="20" data-speed="1500">0</div>
                <div class="cardbox-icon">
                    <i class="material-icons">visibility</i>
                </div>
                <div class="card-details">
                    <h4>Daily visitors</h4>
                    <span>10% Higher than last week</span>
                </div>
            </div>
        </div>
    </div>
    <div class="col-lg-3 col-md-3 col-sm-6 col-xs-12">
        <div class="panel cardbox bg-dark">
            <div class="panel-body card-item panel-refresh">
                <a class="refresh" href="#">
```

```
                <span class="fa fa-refresh"></span>
            </a>
            <div class="refresh-container"><i class="refresh-spinner fa fa-spinner fa-spin fa-5x"></i></div>
            <div class="timer" data-to="@ViewBag.EmailCountCount" data-speed="1500">0</div>
            <div class="cardbox-icon">
                <i class="material-icons">email</i>
            </div>
            <div class="card-details">
                <h4>Total Email</h4>
                <span>10% Higher than last week</span>
            </div>
        </div>
    </div>
</div>
<!-- ./counter Number -->
<!-- Table content -->
<!-- ./Table content -->
<!-- Google Map -->
<div class="col-lg-6 col-md-6 col-sm-6 col-xs-6">

    <div class="card">
        <div class="card-header">
            <i class="fa fa-user-circle-o fa-lg"></i>
            <h2>My Info </h2>
        </div>
        <div class="card-content ">
```

```
<div class="container-fluid ">
  <div class="row">
    <div class="col-lg-4 col-md-4 col-sm-4 col-xs-4"><h4 style="font-family:Arial; font-weight:bold ;color:#17B6A4"> Name :</h4></div>
    <div class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
      <div class=" center">
        <h4 style="font-family:Arial; font-weight:bold ;color:#000"> Mustafa</h4>
      </div>
      <div class="center">
        <img src="~/Content/img/type 3.png" class="img-responsive"  />
      </div>
    </div>
    <div class="col-lg-2 col-md-2 col-sm-2 col-xs-2"></div>
  </div>
  @if (ViewBag.UserType == 3)
  {
    <div class="row">

    <div class="col-lg-4 col-md-4 col-sm-4 col-xs-4"><h4 style="font-family:Arial; font-weight:bold ;color:#17B6A4"> Grade - Section :</h4></div>
    <div class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
      <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12 center">
        <h4 style="font-family:Arial; font-weight:bold ;color:#000"> @ViewBag.GeadeClassSection</h4>
      </div>
      <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12 center">
        <img src="~/Content/img/type 3.png" width="500" height="30" class="img-responsive" />
```

```
            </div>
          </div>
          <div class="col-lg-2 col-md-2 col-sm-2 col-xs-2"></div>
        </div>
      }
      else
      {
        <div class="row">
          <div class="col-lg-4 col-md-4 col-sm-4 col-xs-4"><h4 style="font-family:Arial; font-weight:bold ;color:#17B6A4"> Job Title :</h4></div>
          <div class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
            <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12 center">
              <h4 style="font-family:Arial; font-weight:bold ;color:#000"> Web Application Generator</h4>
            </div>
            <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12 center">
              <img src="~/Content/img/type 3.png" class="img-responsive" width="500" height="30" />
            </div>
          </div>
          <div class="col-lg-2 col-md-2 col-sm-2 col-xs-2"></div>
        </div>
      }
      <div class="row">
        <div class="col-lg-4 col-md-4 col-sm-4 col-xs-4"><h4 style="font-family:Arial; font-weight:bold ;color:#17B6A4"> Department :</h4></div>
        <div class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
          <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12 center">
```

<h4 style="font-family:Arial; font-weight:bold ;color:#000"> @ViewBag.Depertment</h4>

<h4 style="font-family:Arial; font-weight:bold ;color:#17B6A4"> Collage :</h4>

<h4 style="font-family:Arial; font-weight:bold ;color:#000"> @ViewBag.Collage</h4>

<h4 style="font-family:Arial; font-weight:bold ;color:#17B6A4"> University :</h4>

```
                    <h4 style="font-family:Arial; font-weight:bold ;color:#000">
@ViewBag.University</h4>
                </div>
                <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12 center">
                </div>
            </div>
            <div class="col-lg-2 col-md-2 col-sm-2 col-xs-2"></div>
        </div>
    </div>
</div>
<div class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
    <div class="card">
        <div class="card-header">
            <i class="fa fa-map fa-lg"></i>
            <h2>Google Map</h2>
        </div>
        <div class="card-content">
            <!-- Google map -->
            <div class="embed-container">
                <iframe
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d201550.39177761006!2d32.36639818662086!3d37.87842350771473!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x14d08568d615f745%3A0x240dd0fc08060967!2sKonya!5e0!3m2!1sen!2str!4v1511603940130"  height="410" frameborder="0" style="border:0" allowfullscreen></iframe>
                @*<iframe
src="https://www.google.com/maps/embed?pb=!1m14!1m8!1m3!1d25032.16685835845!2d43.2729653215712!3d33.39513700479255!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x155a5c0f96c904cf%3A0xc5536d9aed872b2!2sUniversity+of+Anbar!5e0!3m2!1sen!2str!4v1511291542029"
width="600" height="425" frameborder="0" style="border:0" allowfullscreen></iframe>*@
```

```
            </div>
          </div>
        </div>
      </div>
      <!-- ./Google Map -->
    </div>
    <!-- ./row -->
</div>
<!-- ./cotainer -->
```

_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title Student Management System</title>

    <link rel="shortcut icon" href="~/Content/img/favicon.png"
type="image/vnd.microsoft.icon" />
    <!-- dataTables css -->
    <link href="~/Content/plugins/datatables/dataTables.min.css" rel="stylesheet"
type="text/css" />
    <link href="https://cdn.datatables.net/select/1.2.3/css/select.dataTables.min.css"
rel="stylesheet">
    <!-- jquery-ui css -->
    <link href="~/Content/plugins/jquery-ui-1.12.1/jquery-ui.min.css" rel="stylesheet"
type="text/css" />
    <!-- materialize css -->
    <link href="~/Content/plugins/materialize/css/materialize.min.css" rel="stylesheet">
    <!-- Bootstrap css-->
    <link href="~/Content/bootstrap/css/bootstrap.min.css" rel="stylesheet" />
    <!-- Animation Css -->
    <link href="~/Content/plugins/animate/animate.css" rel="stylesheet" />
    <!-- Material Icons CSS -->
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
    <!-- Font Awesome -->
    <link href="~/Content/plugins/animate/animate.css" rel="stylesheet" type="text/css"
/>
    <link href="~/Content/font-awesome/css/font-awesome.min.css" rel="stylesheet"
type="text/css" />
    <!-- simplebar scroll css -->
```

```
    <link href="~/Content/plugins/simplebar/dist/simplebar.css" rel="stylesheet"
type="text/css" />
    <!-- mCustomScrollbar css -->
    <link href="~/Content/plugins/malihu-custom-scrollbar/jquery.mCustomScrollbar.css"
rel="stylesheet" type="text/css" />
    <!-- custom CSS -->
    <link href="~/Content/dist/css/stylematerial.css" rel="stylesheet">

    <link href="~/Content/plugins/datatable-checkboxes/datatable-checkboxes-
css/dataTables.checkboxes.css" rel="stylesheet" />

    <link href="~/Content/masterDetailStyle.css" rel="stylesheet" />
</head>
<body>

    <div id="wrapper">
        <!--navbar top-->
        @Html.Partial("_Header")
        <!-- Sidebar -->
        <div id="sidebar-wrapper" class="waves-effect" data-simplebar>
            <div class="navbar-default sidebar" role="navigation">
                @Html.Partial("_Menu")
            </div>
            <!-- ./sidebar-nav -->
        </div>
        <!-- sidebar-wrapper -->
        <!-- Page content -->
        <div id="page-content-wrapper">
            <div class="page-content">
                @RenderBody()
                <div class="modal fade" id="myModal" role="dialog">
                </div>
                <div class="modal fade" id="mySubModal" role="dialog">
                </div>
            </div>
        </div>
    </div>
    <!-- ./page-wrapper -->


    <!-- Start Core Plugins
         =====================================================================-->

    <!-- jQuery -->
    <script src="~/Content/plugins/jQuery/jquery-3.2.1.min.js"
type="text/javascript"></script>
    <!-- jquery-ui -->
    <script src="~/Content/plugins/jquery-ui-1.12.1/jquery-ui.min.js"
type="text/javascript"></script>

    <script
src="https://cdn.datatables.net/1.10.16/js/jquery.dataTables.min.js"></script>

    <!-- validate -->
    <script src="~/js/jquery.validate.min.js" type="text/javascript"></script>
    <script src="~/js/jquery.validate.unobtrusive.js" type="text/javascript"></script>
    <!-- notify -->
    <script src="~/js/notify.min.js" type="text/javascript"></script>
```

```
    <!-- Bootstrap -->
    <script src="~/Content/bootstrap/js/bootstrap.min.js"></script>
    <!-- materialize  -->
    <script src="~/Content/plugins/materialize/js/materialize.min.js"
type="text/javascript"></script>
    <!-- metismenu-master -->
    <script src="~/Content/plugins/metismenu-master/dist/metisMenu.min.js"
type="text/javascript"></script>
    <!-- SlimScroll -->
    <script src="~/Content/plugins/slimScroll/jquery.slimscroll.min.js"
type="text/javascript"></script>
    <!-- m-custom-scrollbar -->
    <script src="~/Content/plugins/malihu-custom-
scrollbar/jquery.mCustomScrollbar.concat.min.js" type="text/javascript"></script>
    <!-- scroll -->
    <script src="~/Content/plugins/simplebar/dist/simplebar.js"
type="text/javascript"></script>
    <!-- custom js -->
    <script src="~/Content/dist/js/custom.js" type="text/javascript"></script>
    <script src="~/Content/plugins/monthly/monthly.js" type="text/javascript"></script>
    <!-- End Core Plugins
         ==================================================================-->
    <!-- Start Page Lavel Plugins
         ==================================================================-->
    <!-- dataTables js -->
    <script src="~/Content/plugins/datatables/dataTables.min.js"
type="text/javascript"></script>

    <!-- main js-->
    <script src="~/Content/dist/js/main.js" type="text/javascript"></script>
    <!-- End Theme label Script
         ==================================================================-->
    @*<script src="~/Content/plugins/Datatable-Editor/dataTables.editor.js"></script>
    <script src="~/Content/plugins/Datatable-Editor/dataTables.editor.min.js"></script>*@


    <script
src="https://cdn.datatables.net/select/1.2.3/js/dataTables.select.min.js"></script>

    <script src="~/Content/plugins/datatable-
checkboxes/dataTables.checkboxes.js"></script>
    <script src="~/Content/plugins/datatable-
checkboxes/dataTables.checkboxes.min.js"></script>


    @RenderSection("scripts", required: false)


</body>
</html>
```

_Header.cshtml

```
@{
    IEnumerable<SelectListItem> list = ViewBag.Announcements_list;
    IEnumerable<SelectListItem> Mlist = ViewBag.UMessages_list;
    Layout = null;
}

<nav class="navbar navbar-inverse navbar-fixed-top">
    <!-- Logo -->
    <a class="navbar-brand pull-left" href="">
        <img src="~/Content/dist/img/logo3.png" alt="logo">
    </a>
    <a id="menu-toggle">
        <i class="material-icons">apps</i>
    </a>
    <div class="navbar-custom-menu hidden-xs">
        <ul class="nav navbar-nav navbar-left hidden-xs">
            <li class="top-search "><a href="#search"><i class="material-
icons">search</i></a></li>
            <li>
                <div id="search" class="top-search">
                    <button type="button" class="close">x</button>
                    <form>
                        <input type="search" value="" placeholder="type For searching..."
/>
                        <button type="submit" class="btn">Search</button>
                    </form>
                </div>
            </li>
        </ul>
        <ul class="navbar navbar-right">
            <!--Notification-->
            <li class="dropdown">
                <a class="dropdown-toggle" data-toggle="dropdown">
                    <i class="material-icons">notifications_active</i><span
class="numbers">@ViewBag.Announcements_Count</span>
                </a>
                <ul class="dropdown-menu dropdown-message mCustomScrollbar animated
bounceIn dropdown-notification" data-mcs-theme="minimal">
                    @if (ViewBag.Announcements_Count > 0)
                    {
                        foreach (var item in list)
                        {

                            <li class="list-details">

                                <!-- start notification -->
                                <a href="#"
onclick="PopupForm('@Url.Action("NotifyDetail","Notifications")/' + @item.Value )">
                                    <div class="pro-images pull-left">
                                        <img src="~/Content/dist/img/avatar4.png"
class="img-circle" height="20" width="20" alt="User Image">
                                    </div>
                                    <h5 class="indigo-text">@item.Text</h5>
                                </a>
```

```
                            </li>
                        }

                    }
                    else
                    {
                        <li class="list-details">
                            <center>
                                <h5 style="color: #3f51b5;height:100%;margin-
top:10px">There is no new notifications</h5>
                            </center>
                        </li>
                    }

                </ul>

                <script>

                    function PopupForm(url) {
                        var formDiv = $('<div/>');
                        $.get(url)
                            .done(function (response) {
                                formDiv.html(response);

                                Popup = formDiv.dialog({
                                    autoOpen: true,
                                    resizable: true,
                                    modal: true,
                                    title: 'Notification',
                                    height: 600,
                                    width: 900,
                                    close: function () {
                                        Popup.dialog('destroy').remove();
                                    }

                                });
                            });
                    }

                </script>

                <!-- /.notification -->
                <!--tasks-->
                <!-- /.tasks -->
                <!--tasks-->
            <li class="dropdown">
                <a class="dropdown-toggle" data-toggle="dropdown" href="#">
                    <i class="material-icons">message</i><span
class="numbers">@ViewBag.Messages_Count</span>
                </a>
                <ul class="dropdown-menu dropdown-message mCustomScrollbar animated
bounceIn" data-mcs-theme="minimal">
                    @if (ViewBag.Messages_Count > 0)
                    {
                        foreach (var item in Mlist)
                        {
                            <li class="list-details">
```

```
                                <!-- start notification -->
                                <a href="#"
onclick="PopupForm('@Url.Action("NotifyReplay", "Messages")/' + @item.Value )">
                                    <div class="pro-images pull-left">
                                        <img src="~/Content/dist/img/avatar4.png"
class="img-circle" height="20" width="20" alt="User Image">
                                    </div>
                                    <h5 class="indigo-text">@item.Text</h5>
                                </a>

                            </li>
                        }
                    }
                    else
                    {
                        <li class="list-details">
                            <center>
                                <h5 style="color: #3f51b5;height:100%;margin-
top:10px">There is no new messages</h5>
                            </center>
                        </li>
                    }
                </ul>
                <!-- /.tasks -->
            </li>
            <!--user profile-->
            <li class="dropdown">
                <a class='dropdown-button user-pro' href='#' data-activates='dropdown-
user'>
                    <img src="~/Content/dist/img/avatar5.png" class="img-circle"
height="45" width="50" alt="User Image">
                </a>
                <ul id='dropdown-user' class='dropdown-content'>
                    <li>
                        <a href="@Url.Action("Index","Profile")"><i class="material-
icons">perm_identity</i> View profile</a>
                    </li>
                    <li>
                        <a href="@Url.Action("Index", "Company")"><i class="material-
icons">info_outline</i> About Us</a>
                    </li>
                    <li>
                        <a href="@Url.Action("LogOut", "Profile")"><i class="material-
icons">lock</i> Logout</a>
                    </li>
                </ul>
            </li>
            <!-- /.user profile -->
        </ul>
    </div>
</nav>
```

_Menu.cshtml

```
@using MBA.Core.Entities;
@using MBA.IES.UI.Models;
@{
   int i = 0;
   IList<DetectedUploadsNotify> detectedUploadsNotify = ViewBag.DetectedUploads;
   IList<DetectedTeacherMessageNotify> detectedTeacherMessageNotify = ViewBag.DetectedTeacherMessages;
   IList<DetectedTeacherAnnouncementsNotify> DetectedTeacherAnnouncementsNotify = ViewBag.DetectedAnnouncements;
   int UserType = ViewBag.UserType;
   int detectedUploadsNotifyCount = 0;
   int detectedTeacherMessageNotifyCount = 0;
   int detectedTeacherAnnouncementsNotifyCount = 0;
   if (UserType != 1)
   {
      detectedUploadsNotifyCount = detectedUploadsNotify.Count();
      detectedTeacherMessageNotifyCount = detectedTeacherMessageNotify.Count();
      detectedTeacherAnnouncementsNotifyCount = DetectedTeacherAnnouncementsNotify.Count();
   }
}
@if (UserType == 3)
{
   <nav class="navbar navbar-inverse navbar-fixed-top" style="background-color:#d5d8da; height: 77px; position:relative">
      <div class="navbar-custom-menu hidden-xs center">
         <ul class="navbar ">
            <!--Notification-->
```

```
<li class="dropdown">
    <a class="dropdown-toggle" data-toggle="dropdown" style="height: 75px;">
        <i class="material-icons">notifications_none</i><span class="secondnumbers">@detectedTeacherAnnouncementsNotifyCount</span>
    </a>
    <ul class="dropdown-menu dropdown-message mCustomScrollbar animated bounceIn dropdown-notification" data-mcs-theme="minimal">
        @if (detectedTeacherAnnouncementsNotifyCount > 0)
        {
            foreach (var item in DetectedTeacherAnnouncementsNotify)
            {

            @: <li class="list-details">
                @:<a href="#" onclick="PopupForm('@Url.Action("NotifyDetail", "StudentAnnouncements")/' + @item.Id)">
                    <div class="tasks pull-left">
                        <i class='fa fa-bullhorn' aria-hidden='true'></i>
                    </div>
                    <p style="font-weight:bold;font-size:12px; color:#26a69a">@item.Title</p>
                    <p>@item.LessonName</p>
                    <p style="">@item.EmployeeName</p>
                        <span class="badge red">@item.Date</span>
                        @: </a>
                        @:</li>
                }
        }
        else
        {
```

```
<li class="list-details">
    <center>
        <h5 style="color: #3f51b5;height:100%;margin-top:10px">There is no new notifications</h5>
    </center>
</li>
}
</ul>
<!-- /.notification -->
<!--tasks-->
</li>
<li class="list-details dropdown">
<a class="dropdown-toggle" data-toggle="dropdown" href="#" style="height: 75px;">
<i class="material-icons">mail_outline</i><span class="secondnumbers">@detectedTeacherMessageNotifyCount</span>
</a>
<ul class="dropdown-menu aaaa dropdown-message mCustomScrollbar animated bounceIn" data-mcs-theme="minimal">
@if (detectedTeacherMessageNotifyCount > 0)
{
foreach (var item in detectedTeacherMessageNotify)
{
@: <li class="list-details">
@:<a href="#" onclick="PopupForm('@Url.Action("NotifyReplay", "StudentMessages", new { @id = item.TeacherMessageId, @masterId = item.Id })')">
<div class="tasks pull-left">
<i class="fa fa-file-text-o blue-grey"></i>
</div>
```

```
                    <p>@item.Title</p>
                    <p>@item.LessonName</p>
                    <p style="">@item.Sender</p>
                    <span class="badge red">@item.Date</span>
                    @: </a>
                  @: </li>
            }
              }
              else
              {
              <li class="list-details">
                <center>
                  <h5 style="color: #3f51b5;height:100%;margin-top:10px">There is no new messages</h5>
                </center>
              </li>
              }
            </ul>
          </li>
          <!--tasks-->
          <li class="dropdown">
            <a class="dropdown-toggle" data-toggle="dropdown" href="#" style="height: 75px;">
              <i class="material-icons">file_upload</i><span class="secondnumbers">@detectedUploadsNotifyCount</span>
            </a>
            <ul class="dropdown-menu bbbb dropdown-message mCustomScrollbar animated bounceIn" data-mcs-theme="minimal">
@if (detectedUploadsNotifyCount > 0)
```

```
{
                foreach (var item in detectedUploadsNotify)
                {
                   @:<li class="list-details">
                     @:<a href="#" onclick="PopupForm('@Url.Action("NotifyUploads", "StudentUploads")/' + @item.DetailId)">
                   <div class="tasks pull-left">
                   <i class='fa fa-upload' aria-hidden='true'></i>
                </div>
                        <p style="font-weight:bold;font-size:12px; color:#26a69a">@item.Title</p>
                            <p>@item.LessonName</p>
                            <p style="">@item.EmployeeName</p>
                            <span class="badge red">@item.UploadDate</span>
                            @: </a>
                            @:</li>
                        }
}
            else
            {
<li class="list-details">
   <center>
      <h5 style="color: #3f51b5;height:100%;margin-top:10px">There is no new uploads</h5>
   </center>
</li>
            }
            </ul>
            <!-- /.tasks -->
```

```
            </li>
            <!--user profile-->
          </ul>
        </div>
    </nav>
}
else if (UserType == 2)
{
    @:<nav class="navbar navbar-inverse navbar-fixed-top" style="background-color:#d5d8da; height: 77px; position:relative">
        @: <div class="navbar-custom-menu hidden-xs center">

            <ul class="navbar ">
                <li></li>
                <!--Notification-->
                <li class="list-details dropdown">
                    <a class="dropdown-toggle" data-toggle="dropdown" href="#" style="height: 75px;">
                        <i class="material-icons">mail_outline</i><span class="secondnumbers">@detectedTeacherMessageNotifyCount</span>
                    </a>
                    <ul class="dropdown-menu  dropdown-message mCustomScrollbar animated bounceIn" data-mcs-theme="minimal">
                        @if (detectedTeacherMessageNotifyCount > 0)
                        {
                            foreach (var item in detectedTeacherMessageNotify)
                            {
                                @: <li class="list-details">
```

```
                    @:<a href="#" onclick="PopupForm('@Url.Action("NotifyReplay",
"TeacherMessages", new { @id = item.TeacherMessageId, @masterId = item.Id })')">
                        <div class="tasks pull-left">
                            <i class="fa fa-file-text-o blue-grey"></i>
                        </div>
                        <p>@item.Title</p>
                        <p>@item.LessonName</p>
                        <p style="">@item.Sender</p>
                        <span class="badge red">@item.Date</span>
                        @: </a>
                    @: </li>
            }
                }
                else
                {
                    <li class="list-details">
                        <center>
                            <h5 style="color: #3f51b5;height:100%;margin-top:10px">There is no
new messages</h5>
                        </center>
                    </li>
                }
            </ul>
        </li>
    </ul>
    @:</div>
@:</nav>
}
```

```
<div class="sidebar-nav navbar-collapse" >

    <ul class="nav" id="side-menu">

        <li class="text-center active-link" style="color:#17B6A4;border-bottom:1px solid #17B6A4"><a><i class="material-icons" style="padding: 0 1px 3px 0px!important;color:#17B6A4">apps</i> Menu</a></li>

        <li class="active-link"><a href="@Url.Action("Index" , "Home")"><i class="material-icons">dashboard</i>Home</a></li>

        @{
            List<MenuItems> menus = new List<MenuItems>();
            if (Session["SiteSession"] != null)
            {
                SiteSession siteSession = (SiteSession)Session["SiteSession"];
                menus = siteSession.menus;
            }
            foreach (MenuItems m in menus)
            {
                if (m.ChildMenuItems.Count() == 0)
                {
                    <li class="active-link"><a href="@Url.Action(m.Action , m.Controller)"><i class="material-icons">@m.MenuIcon</i>@m.MenuItem</a></li>
                }
                else
                {
                    <li class='has-sub'>
                        <a href="#"> <i class="material-icons">@m.MenuIcon</i>@m.MenuItem <span class="fa arrow"></span></a>

                        @if (m.ChildMenuItems.Count() > 0)
        { <ul>
```

```
                @foreach (MenuItems sm in m.ChildMenuItems)
                {
                    i++;
                    if (i < m.ChildMenuItems.Count)
                    {
                        <li class=""><a href="@Url.Action(sm.Action , sm.Controller)"><i class="material-icons">@sm.MenuIcon</i>@sm.MenuItem</a></li>
                    }
                    else
                    {
                        <li class="last"><a href="@Url.Action(sm.Action , sm.Controller)"><i class="material-icons">@sm.MenuIcon</i>@sm.MenuItem</a></li>
                    }
                }
            </ul>
                    }
                </li>
            }
        }
    </ul>
    <!-- ./sidebar-nav -->
</div>
<script>
    function PopupForm(url) {
        var formDiv = $('<div/>');
        $.get(url)
            .done(function (response) {
```

```
                formDiv.html(response);
                Popup = formDiv.dialog({
                    autoOpen: true,
                    resizable: true,
                    modal: true,
                    title: 'Notification',
                    width: 900,
                    close: function () {
                        Popup.dialog('destroy').remove();
                    }
                });
            });
        }
    </script>
<script>
    function SubPopup(url) {
        $('#mySubModal').load(url);
    }
</script>
```