MVC FRAMEWORK

INDUSHREE H S
Information Science And Technology
Presidency University
Bangalore,India
indushrees99@gmail.com

Abstract—The features and the capabilities of web applications are growing rapidly, and the complexities and difficulties of web applications engineering are also growing in parallel. If the architectural formalism of these advanced web applications is well realized, the complexities could be understood, thus the difficulties could be reduced. Model-View-Controller (MVC) has been recognized as a well-formed architectural style, and has been widely used in web applications engineering in various forms of implementations. These MVC implementations are heavily dependent on specific set of technologies and/or some other facts; hence, they do not provide an abstract realization to be used in a wider range of web application engineering. We propose an implementation of MVC in more abstract form, which – we think – will increase the realization of the advanced web applications, thus lower the
engineering complexities and difficulties of web applications. We believe that this implementation is more applicable in a wider range of environments and technologies, and will upturn the architectural properties like performance and modifiability. Based on this implementation we introduce an MVC based architectural style for web applications.

I. TIINTRODUCON

The **.NET Framework** is a software development framework developed by Microsoft that provides a runtime environment and a set of libraries and tools for building and running applications on Windows operating systems. The framework includes a variety of programming languages, such as C#, F#, and Visual Basic, and supports a range of application types, including desktop, web, mobile, and gaming applications.

1.The .NET Framework includes two main components: the Common Language Runtime (CLR) and the .NET Framework Class Library. The CLR is responsible for managing the execution of code written in any of the supported languages, while the class library provides a large set of pre-built functions and classes that can be used to create a wide range of applications.

2.Another advantage of the .NET Framework is its support for a variety of application types. The framework includes libraries and tools for creating desktop, web, mobile, and gaming applications, which makes it a versatile choice for developers working on a wide range of projects.

2. Problem and Motivation

The web applications have evolved into more advancedsystems and their complexity has grown significantly , where diverse types of components are integrated into various ways in modern web applications, therefore causing difficulties in understanding the architectural formalism of
them. This setting affects the engineering processes of the web-based systems in a negative manner. To address the related issues and support web engineering, numerous concepts, and TTs have been introduced. These supporting
artifacts come with additional learning curves along with their pros and cons.
However, the foundation of these advanced web
applications is still laid on the Client-Server (C-S) model,

II. METHODOLOGY

A literature survey was conducted to gain the domainknowledge of the areas of the web applications, software architecture and architectural styles, conceptual abstraction
and TTs independency, MVC, and the TTs used to develop MVC based web applications.
It was noted that there is lack of literature for TTs
independency and the concept abstraction. Therefore to gain that related knowledge through experiencing the

utilization of available TTs into MVC based web development – towards gaining empirical evidence – a series of experiments
was conducted. The experiments were prototype based and conducted in an incremental manner. Facts learned in literature were tested in early iterations, in the direction of identifying the bottlenecks and issues, then solutions for the
identified problems were tested in later iterations. Identified solutions were continuously refined to verify that they do not conflict with the artifacts found in later iterationssame as the “Heading 1” style but without numbering.

3.Modules in MVC

MVC Framework:

The MVC (Model, View, Controller) framework is an architectural/design pattern that separates an application into three main logical components Model, View, and Controller. Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other. MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects.

The main advantage of using the MVC framework is that it provides a clear separation of concerns between the different components of the application. This makes the application easier to develop, test, and maintain, as each component can be developed independently of the others. It also makes the application more scalable and extensible, as new functionality can be added to the application without affecting the existing code.

Another benefit of using the MVC framework is that it supports the use of templates and partial views, which can be used to create reusable UI components that can be shared across multiple pages of the application. This can help to improve the consistency and maintainability of the application's user interface.

The MVC framework includes the following 3 components:

1) Controller

2) Model

3) View

3.2 Controller:

The controller is the component that enables the interconnection or the coordinator between the View and the Model. The controller is responsible for handling user requests and coordinating the interaction between the Model and View components of the application. When a user makes a request to the application, the controller receives that request and determines which action needs to be taken based on the request parameters and other contextual information. The controller then interacts with the Model component to perform any necessary data retrieval or manipulation, and passes the results of those operations back to the View component for rendering and display to the user. In addition to handling user requests, the controller also typically handles any exceptions or errors that occur during the processing of those requests, and may perform other tasks such as authentication and authorization checks.

One important aspect of the controller is that it should be lightweight and focused on performing its coordinating role, rather than containing complex business logic or other application-specific functionality. This helps to keep the application maintainable and modular, as it allows for easier testing, reuse, and evolution of the individual components.

3.3 Model:

The Model component is responsible for managing the application's data and business logic. It can represent either the data being transferred between the View and Controller components or any other data that is related to the business logic of the application.One of the key roles of the Model component is to interact with the database or other data sources to retrieve and manipulate data. This involves executing queries, updates, and other database operations to retrieve or modify data as needed. The Model component may also perform data validation

and other operations to ensure that the data is consistent and accurate.

In addition to database interactions, the Model component may also contain other business logic related to the application's functionality. For example, it may perform calculations, generate reports, or perform other operations that are required by the application's requirements. One important aspect of the Model component is that it should be designed to be modular and extensible, so that it can be adapted to different types of data sources or business logic requirements. This may involve implementing interfaces, abstract classes, or other design patterns that allow for easy customization and extension of the Model component.

There are 4 components of the Model in MVC :

- Entity Model
- View Model
- View Data
- Repository

3.3.1 Entity Model:

It contains all the class object description of all the instances that are created for the given screen. It also maps the database tables. It is used to simplify the process of accessing and manipulating data in the data store, by providing a high-level abstraction of the data and its relationships.

In the context of the MVC framework, an Entity Model typically refers to the use of the Entity Framework, which is a popular Object-Relational Mapping (ORM) framework provided by Microsoft. The Entity Framework enables developers to create a conceptual model of their data store using Entity Data Models (EDMs), which are representations of the entities and relationships in the data store.

The Entity Framework then generates the necessary code to access and manipulate the data in the data store, based on the EDM. This can greatly simplify the development process, as developers can work with objects and classes rather than writing low-level SQL queries.

3.3.2 View Model:

ViewModel is an intermediary between the Model and the View components. Its purpose is to provide a simplified, domain-specific representation of the data that the View needs to display, while also keeping the Model decoupled from the View. In other words, the ViewModel is a specialized class that encapsulates the data that the View needs to display, and provides a way for the View to access that data without having direct knowledge of the Model. This can be useful for a number of reasons, such as:

- Separation of concerns: By using a ViewModel, the View can be decoupled from the Model, allowing the two components to be developed independently and reducing the risk of breaking changes when modifying one component.
- Data transformation: The ViewModel can perform data transformation or formatting to ensure that the data is displayed in the way that the View requires.
- Domain-specific representation: The ViewModel can provide a simplified, domain-specific representation of the data that is tailored to the needs of the View, making it easier for the View to work with the data.

3.3.3 View Data:

View Data is a way for the Controller component to pass data to the View component. It provides a

mechanism for the Controller to share data with the View without having to expose the Model directly to the View.

View Data can be used to pass any type of data from the Controller to the View, such as strings, numbers, objects, or collections. The data is typically stored in a dictionary-like structure called ViewData or ViewBag, which can be accessed in the View using Razor syntax or other templating languages. One of the benefits of using View Data is that it allows the Controller to customize the data that is sent to the View based on the specific needs of the View. For example, the Controller could use View Data to pass a list of products to the View, but then filter or sort the list before it is displayed in the View.

Another benefit of View Data is that it allows the Controller to provide additional context or metadata about the data being displayed in the View. For example, the Controller could use View Data to pass information about the current user, the current page, or other contextual information that may be useful for rendering the View.

3.3.4 Repository:

Repository is a design pattern that provides an abstraction layer between the data access layer and the rest of the application. The Repository pattern is commonly used in web applications to simplify the process of accessing and managing data from a database or other data source. The basic idea behind the Repository pattern is to encapsulate the logic for accessing data into a single class or set of classes. This allows the rest of the application to work with a consistent and well-defined API, rather than having to deal directly with the complexities of data access.

Repository typically provides a set of methods for querying and manipulating data. These methods can include basic CRUD (Create, Read, Update and Delete) operations, as well as more complex queries and transactions. The Repository also abstracts away the specific implementation details of the data access layer, such as the underlying database or ORM (Object-Relational Mapping) framework being used.

Using a Repository can provide a number of benefits for a web application, including:

- Improved code maintainability: By encapsulating data access logic in a single place, it becomes easier to modify and test the code.
- Better separation of concerns: The Repository pattern helps to separate the data access layer from the rest of the application, making it easier to understand and modify both parts of the application independently.
- Improved scalability: By providing a consistent and well-defined API for accessing data, the Repository pattern makes it easier to optimize and scale the application's data access layer.
- Improved security: By centralizing the logic for accessing and manipulating data, the Repository pattern can help to prevent common security issues such as SQL injection attacks.

3.4 Views:

View is a component responsible for displaying data to the user and capturing user input. Views are responsible for rendering the user interface, which the user interacts with The View component is decoupled from the business logic, which is handled by the Model and Controller components. Views typically do not perform any data processing or manipulation, but rather display data retrieved from the Model or user input captured by the Controller.

Views can take many different forms depending on the application requirements. They can be simple HTML, CSS and JavaScript pages rendered by a web server, desktop application windows, mobile app screens, or any other interface used to interact with the user.

In the MVC pattern, the View is connected to the Model and Controller through a series of well-defined interfaces. The Model provides the data that the View displays, and the Controller is responsible for updating the Model based on user input received through the View.

Views can also implement a variety of design patterns, such as the ViewModel pattern, which provides a layer of abstraction between the View and Model components. This helps to separate the presentation logic from the business logic, making it easier to maintain and extend the application over time.

ACKNOWLEDGMENTS

Model–View–Controller is a popular software pattern• used to break up the logic of your application into three different components.While the MVC pattern was initially used in desktop applications, it became popular to use in web applications during the late 1990's.The Model is responsible for the data logic behind the application.The View is what the user sees and interacts with in the application.The Controller acts as the brains behind the application and communicates with the Model and View.Web frameworks that use the MVC pattern include, Ruby on Rails, ASP.NET MVC, Laravel, and Angular.

REFERENCES

BALANCED ABSTRACT WEB-MVC STYLE: AN ABSTRACT MVC IMPLEMENTATION FOR WEB-BASED APPLICATIONS
May 2017
DOI:10.5176/2251-3043_5.3.375.

Make sure to remove all placeholder and explanatory text from the template when you add your own text. This text should not be here in the final version!