International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 05 | May - 2023 SJIF 2023: 8.176 ISSN: 2582-3930

Comprehensive look of the ASP.NET MVC

Sabah Sarwari, V Gnanesh, Tejaswini HP
Department of Computer Science and Engineering

Presidency University, Bangalore

Instructor:

Amirtha Preeya V Asst. Prof-CSE, Mr. Ragavendra M Devadas Asst. Prof-CSE, Mr. Laksmisha.S. K Asst. Prof-CSE
Department of Computer Science and Engineering

Presidency University, Bangalore

ABSTRACT

This research paper presents an in-depth introduction and examination of the ASP.NET Model-View-Controller (MVC) framework. Microsoft's ASP.NET is a well-known web development platform. This article examines the MVC framework in ASP.NET's fundamental principles, features, and implementation details, emphasising its role in fostering division of concerns, maintainability, and extensibility. It also looks at real-world examples and explores recommended practices for using the MVC framework to create resilient and scalable online applications.

Index Terms- MVC Architecture, ASP.NET, Controllers, Models, Views

Introduction

a. The MVC pattern's history and relevance in web development:

The introduction presents a historical framework for web development as well as the necessity for architectural patterns. Traditional techniques to web development, such as Web Forms, sometimes resulted in tightly connected code that was hard to maintain and verify. By advocating a clear separation of responsibilities, the MVC design arose as a response to these difficulties.

b. Outline of ASP.NET and its MVC framework support:

Microsoft's ASP.NET is a renowned web development framework. It began with Web Forms as the main development format, but later included the MVC framework as an option. The MVC architectural style is used by ASP.NET MVC to give developers with a more organised and modular method to design online applications.

MVC Architecture

a. Model, View, and Controller Component Explanation:

In the MVC framework, the Model comprises the application and business logic information and business logic, the View handles presentation and user interface, and the Controller oversees the interaction between the Model and View. This separation of responsibilities enables each component to be developed and tested independently, boosting reuse of code and maintainability.

b. Component relationships and interactions:

The Model, View, and Controller communicate via well-defined interfaces. The Controller accepts user input, executes relevant Model operations, and chooses the right View to present the new data. This separation allows each component to expand independently of the others, resulting in simpler to manage and extensible codebases.

c. Advantages of utilising the MVC Pattern in Web Application Development:

Studies as well as industry examples have revealed various advantages of utilising the MVC pattern in web application development. MVC improves code maintainability by separating concerns, allowing developers to focus on particular elements of the programme. MVC's modular structure allows for better code organisation and reuse, which reduces development effort as well as time. Furthermore, the division of the user interface in the View component simplifies the deployment of responsive design and device support.

For example, Li and Li (2018) conducted a study that contrasted the maintainability of MVC-based apps against conventional Web Forms applications and discovered that the MVC method resulted in superior code maintainability, lowering the time and effort necessary for bug repairs and feature upgrades. Similarly, industry leaders such as Stack Overflow and GitHub have used the MVC paradigm to create scalable and maintainable online applications.

ASP.NET MVC Key Concepts

a. Routing and URL Mapping:

ASP.NET MVC includes a customizable routing mechanism that connects URLs to the relevant Controller operations. Developers may have clear and understandable URLs that match the structure of the application by designing custom routes. The routing system supports a variety of patterns and limitations, making it possible to create SEO-friendly URLs and RESTful APIs. Consider the URL "https://example.com/products/123" as an example. This URL may be mapped to the "Product Controller" and its "Details" action by supplying the product ID as a parameter to the routing system.

b. Action ways and controllers:

Action methods serve as the foundation for MVC Controllers. They handle user requests, communicate via the Model layer, and decide which View to render. The model of MVC automatically translates URLs to matching Controller actions using naming conventions, giving a simple and uniform approach to request management. Controllers are critical in controlling the data and control flow in an MVC application. They are in charge of processing user input, calling the relevant business logic, and producing the data for display in the View.

c. Views and Razor syntax:

Views are in charge of designing the user interface and displaying data to the user. Views are commonly defined in ASP.NET MVC using Razor syntax, which blends HTML markup with server-side code. Razor generates dynamic content in a compact and expressive manner, allowing for easy integration of server-side functionality into view templates. Razor syntax enables programmers to write server-side code straight in HTML markup, improving view readability and maintainability. Developers, for example, may simply continue iterating over a set of data and build a table dynamically using Razor syntax.

d. Data binding and model validation:

ASP.NET MVC includes powerful data binding features that enable for the automated translation of user input from forms to Model attributes. This feature streamlines the processing of form submissions by removing the requirement for manually extraction of information and validation. Model validation is an essential component of ASP.NET MVC because it ensures the accuracy and safety of user-submitted data. The framework has validation characteristics that may be used with Model properties to impose limitations like mandatory fields, length limits, and data type validation.

e. Attribute-based programming and filters:

Filters enable the addition of cross-cutting behaviour to MVC applications. They let developers to inject arbitrary logic prior to, during, or following Controller actions or views are executed. Filters in ASP.NET MVC include Permission Filters, Action Filters, Result Filters, and Exception Filters. Filters allow attribute-based programming, which specifies specific behaviours by applying attributes to controllers or actions methods. By encapsulating similar functionality under characteristics that can be readily applied to other portions of the programme, this method increases code reusability and modularity.

ASP.NET MVC Framework Implementation

a. Project Installation and Configuration:

Selecting the proper project template, establishing dependence, and specifying the project structure are all steps in creating an ASP.NET MVC project. Project templates, such as those provided by Visual Studio, automate most of the start-up process, including the generation of necessary files and directories. An MVC project's setup comprises providing routing rules, configuring the dependency injection (DI) containers, and controlling data access, authentication, and permission settings. Developers may customise the behaviour of the MVC framework to meet their individual needs by correctly setting the project.

b. Designing models, views, and controllers:

Models are created by creating classes that represent the application's data elements and encapsulate the business logic. To get and alter data, models can communicate with databases, websites, or other data sources.

Views are in charge of displaying data to users and gathering user input. They are usually made up of HTML markup with embedded code on the server written in Razor syntax. Views are defined by developers depending on the data needs and the intended user interface.

In an MVC application, controllers act as the core control point. They accept user requests, analyse them by performing relevant Model operations, and select which View to render. Controller classes are created by developers, and action methods are defined to handle particular user interactions.

c. Routing setup and customization:

The routing mechanism in ASP.NET MVC is extremely versatile and adaptable. Custom routes may be defined by developers to map certain URLs to Controller action. Custom routes generate clean and understandable URLs, which improves total user experience and search engine optimisation (SEO).

Routes may be configured using attributes or by declaring them in the route table. Developers can define routing information from within the Controller or action method using attributes such as the [Route] element. This method simplifies the definition of routing rules and aids in the separation of routing and business logic. Route restrictions can also be used by developers to evaluate and limit the value of route parameters. Constraints enable extremely fine surveillance of URL patterns and guarantee that only legitimate URLs are used.

d. Processing form submissions and user input:

ASP.NET MVC's model binding technique simplifies the processing of form submissions and user input. Based on naming standards, model binding automatically links form data to matching Model attributes. This feature reduces the need for developers to manually extract form values, allowing them to focus on analysing the provided data.

Model validation is critical for assuring the accuracy and safety of user input. Validation attributes such as [Required], [StringLength], and [RegularExpression] are integrated into ASP.NET MVC and may be applied to Model properties. When a form is uploaded, the framework automatically validates it based on these attributes and notifies the user if there are any validation issues.

Developers may leverage JavaScript modules or frameworks to incorporate methods like client-side validation to improve user input handling even further. This method evaluates input from users on the client side prior submitting the form, giving the user fast response and eliminating needless server round-trips.

e. Using the MVC framework's data access techniques:

Access to data is a vital component of web application development. ASP.NET MVC enables developers to communicate with databases, web services, and other data sources via a variety of data access strategies.

Using an Object-Relational Mapping (ORM) framework such as Entity Framework (EF) is a popular option. EF facilitates database interaction through offering an abstraction layer that translates entities from databases to Model objects. It includes functions like as automated CRUD operations, query creation, and change tracking.

Another method is to separate the data access layer from the rest of the programme by utilising repositories. Repositories isolate the mechanism for retrieving and storing data, allowing the MVC elements and the underlying data storage to be loosely coupled.

Dependency injection (DI) is a critical practise in ASP.NET MVC, allowing component decoupling and boosting testability and maintainability. DI containers, such as the built-in container in ASP.NET Core or third-party libraries like Autofac or Ninject, make dependency management and service injection into Controllers or other components easier.

Overall, ASP.NET MVC gives developers the freedom to select the best data access mechanisms for their projects. Developers may design strong and maintainable data access layers by utilising frameworks like as Entity Framework and applying repositories patterns with dependency injection.

Advanced Concepts and Best Practises

a. Utilising view models to improve data encapsulation:

View models are frequently used in complicated systems to encapsulate data and logic for presentation relevant to a single view. View models are class that represent the data required for a given view and may have extra attributes or methods to improve the functionality of the view.

Developers may achieve greater separation of concerns by using view models, which contain the data needs and presentation logic particular to a view. This method helps to keep the Model and View separate, reducing tight coupling and enhancing maintainability.

Furthermore, view models allow data from different models to be mapped into a single coherent object, removing the need to reveal the complete domains model to the view. This encapsulation lowers the danger of revealing important or superfluous data while also improving security.

b. Using partial views and layout pages to implement reusable components:

ASP.NET MVC includes tools for constructing reusable UI components like partial views and design pages. Partial views are views that may be displayed inside other views, which promotes reuse of code and modular architecture.

The standard structure and layout of various views are defined by layout pages. They usually feature the similar HTML structure, navigation menus, and other aspects that are shared. Developers may use layout pages to ensure uniformity across multiple views, eliminating code duplication and improving maintainability. Reusable components, such as partial views and layout pages, encourage code reuse, increase developer efficiency, and provide a uniform user experience across the programme.

c. ASP.NET MVC testing and test-driven development (TDD):

Testing is an important part of software development since it ensures the application's accuracy and dependability. Testing methodologies supported by ASP.NET MVC include unit testing, integration testing, and testing for acceptance.

Unit testing enables developers to isolate and test specific components such as Controllers and Model classes. It guarantees that each component behaves as planned and aids in the identification and resolution of issues early in the design process.

Integration testing examines the interplay of several components to ensure that they perform properly together. This form of testing is very beneficial for validating the integration of Controllers, Models, and Views.

Acceptance testing checks the application's behaviour from the standpoint of the user. It entails creating tests that imitate user interactions and ensure that the programme works as planned.

TDD is a development process in which tests are created prior to the implementation code. This strategy aids in the development process by verifying that the application satisfies the requirements and lowering the possibility of bugs being introduced.

Developers may improve the quality and reliability of ASP.NET MVC applications by using testing methodologies like as unit testing, integration testing, and acceptance tests.

In comparison to Other Web Development Frameworks

a. In comparison to other common frameworks (e.g., Web Forms, Razor Pages):

When selecting a web development framework, compare MVC to alternative frameworks often used in the ASP.NET environment, such as Web Forms and Razor Pages. Each strategy has advantages and disadvantages, and the choice is determined by the project's individual requirements.

The typical technique in ASP.NET is Web Forms, which employs a component-based paradigm and focuses on abstracting the difficulties of web development. It offers a comprehensive set of server controls, event-driven programming, and automated ViewState management. Web Forms are ideal for quick application development, but they can result in tightly connected code and are less adaptable in some cases.

Razor Pages, which became available in ASP.NET Core, are a simpler programming approach for developing online applications. Razor Pages combines the advantages of MVC and Web Forms, enabling developers to create pages with an emphasis on clarity and separate concerns. Razor Pages are best suited for simple apps or when collaborating with a small group.

Developers may make educated selections by comparing MVC to various frameworks based on aspects such as reliability, flexibility, efficiency, and knowledge with the development approach.

b. Assessing the strengths and limitations of each approach:

It is critical to assess MVC's strengths and shortcomings when compared to other frameworks. MVC encourages explicit separation of responsibilities and modular architecture, which improves maintainability and extensibility. It gives you more influence over the HTML output and is ideal for sophisticated applications with a wide range of needs.

Web Forms, on the other hand, provides a component-based paradigm that simplifies some parts of web development and is ideal for quick application development scenarios. However, it can lead to tightly connected code that is less adaptable in some cases.

Razor Pages offer a streamlined programming paradigm that combines the benefits of MVC with Web Forms, providing a simple solution for designing lightweight apps or when collaborating with a small team. It may, however, be unsuitable for sophisticated applications with an important level of segregation between the Model and the View.

The decision between MVC, Web Forms, and Razor Pages is determined by the individual project needs, the skill of the development team, and factors such as reliability, performance, and scalability.

Conclusion

a. Key discoveries and contributions summarised:

The Model-View-Controller (MVC) architecture in ASP.NET was thoroughly investigated in this research study. It has presented a thorough examination of the MVC design, its essential ideas, and the advantages it provides in web application development. The presentation went over the MVC framework's ASP.NET implementation specifics, including as routing, controllers, views, data access, and advanced subjects like view models and reusable components. It has also compared MVC to other frameworks for developing websites and evaluated the advantages and disadvantages of each technique.

b. Consequences and future directions:

Because of its emphasis on separation of responsibilities, maintainability, and extensibility, the MVC framework in ASP.NET remains a popular choice for online application development. Further research and development efforts might focus on improving the efficiency and scaling of MVC-based applications, researching new features and improvements, and examining the incorporation of emerging technologies as technology progresses.

c. Closing remarks:

Finally, the ASP.NET MVC framework offers developers with a powerful and versatile architectural pattern for developing web applications. Developers can gain from code reliability, modularity, and testability by using MVC. The extensive range of tools provided by ASP.NET MVC, including as routing, data binding, and filters, aid in the development process and enable the production of scalable and responsive online applications.

Overall, the ASP.NET MVC framework provides a strong and tried-and-true approach to online application development, allowing developers to create robust, maintainable, and extendable solutions. Developers may design high-quality web apps that satisfy the needs of contemporary software development by following best practises and using the MVC pattern.

Reference

1) Leff, A., & Rayfield, J. T. (2001). Web-application development using the model/view/controller design pattern. In Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International (pp. 118-127). IEEE.
2) Pro ASP.Net MVC 5 book by Adam Freeman
3) Jailia, M., Kumar, A., Agarwal, M., & Sinha, I. (2016, November). Behavior of MVC (Model View Controller) based Web Application developed in PHP and. NET framework. In ICT in Business Industry & Government (ICTBIG), International Conference on (pp. 1-5). IEEE.
4) Gupta P, Govil MC (2010) MVC Design pattern for the multi framework distributed applications using XML, spring and struts framework. International Journal on Computer Science and Engineering 2(4): 1047-1051.