Application Development using ASP.NET

H.Asela
Department of Electrical and Information Engineering
University of Ruhuna

MVC Architecture

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller.

It is a design pattern is a structure for keeping display and data separate to allow each to change without affecting the other.

It is provides the fundamental pieces for designing a programs for desktop or mobile, as well as web applications.

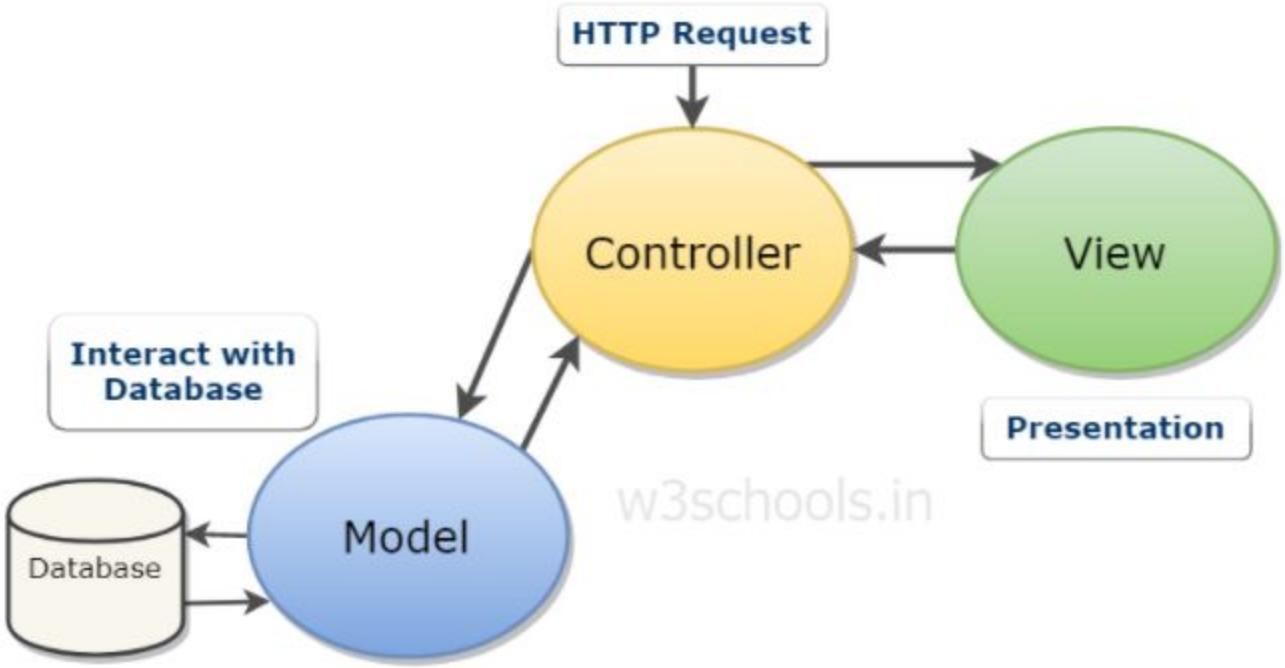


Fig: MVC Architecture

Model

The Model encloses the clean application related data. But the model does not deal with any logic about how to present the data.

View

The View element is used for presenting the data of the model to the user. This element deals with how to link up with the model's data but doesn't provide any logic regarding what this data all about or how users can use these data.

Controller

The Controller is in between the model and the view element. It listens to all the incident and actions triggered in the view and performs an appropriate response back to the events.

Benefits of MVC Architecture

Logical clustering of related acts on any controller can be achieved through MVC.

Various developers can work at the same time on different parts the same application-controller, model, and the views part.

Models can have numerous views.

Entity Framework Core

Entity Framework (EF) Core is a lightweight, extensible, open source and cross-platform version of the popular Entity Framework data access technology.

EF Core can serve as an object-relational mapper which:

- Enables .NET developers to work with a database using .NET objects.
- Eliminates the need for most of the data-access code that typically needs to be written.

DBContext Class

A DbContext instance represents a combination of the Unit Of Work and Repository patterns such that it can be used to query from a database and group together changes that will then be written back to the store as a unit.

```
public class StudentContext : DbContext
{
    public StudentContext(DbContextOptions options) : base(options) { }
    public DbSet Students{ get; set; }
}
```

Add Service

```
services.AddDbContext(opt =>
opt.UseSqlServer(Configuration.GetConnectionString("StudentContext")))
```

Add Service

```
services.AddDbContext(opt =>
opt.UseSqlServer(Configuration.GetConnectionString("StudentContext")))
```

Configurations

In ASP.NET Core the configuration system is very flexible, and the connection string could be stored in appsettings.json, an environment variable, the user secret store, or another configuration source.

Connection string provides the information that a provider needs to communicate with a particular database.

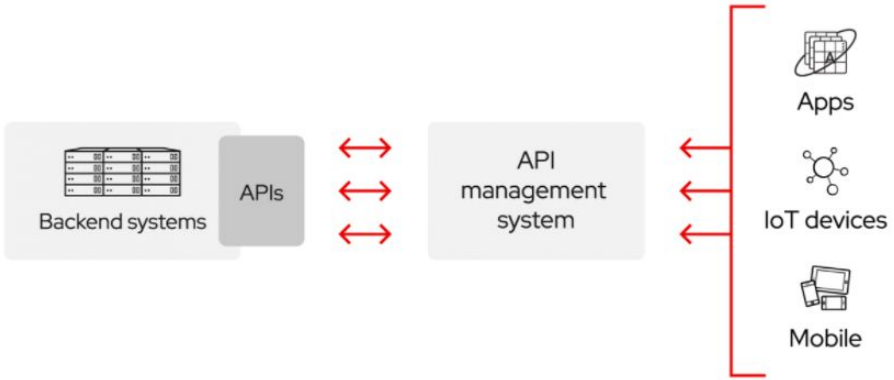
It includes parameters such as the name of the driver, Server name and Database name , as well as security information such as username and password.

```
"ConnectionStrings":
{
    "TestContext": "Server=(localdb)\\mssqllocaldb;
    Database=TestContext;
    Trusted_Connection=True;
    MultipleActiveResultSets=true"
}
```

API

An API is a set of definitions and protocols for building and integrating application software. API stands for application programming interface.

APIs let your product or service communicate with other products and services without having to know how they're implemented.



REST API

REST (Representational state transfer) is an API that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services.

Each URL is called a request while the data sent back to is called a response.

The endpoint (or route) is the url you request for. It follows this structure:

Root-endpoint / ?

HTTP

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems.

This is the foundation for data communication for the World Wide Web (i.e. internet) since 1990.

HTTP is a generic and stateless protocol which can be used for other purposes as well using extensions of its request methods, error codes, and headers.

HTTP Requests

An HTTP client sends an HTTP request to a server in the form of a request message.

The request method indicates the method to be performed on the resource identified by the given Request-URI.

The method is case-sensitive and should always be mentioned in uppercase.

GET

The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.

POST

A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.It is used to send data to a server to create/update a resource.

PUT

PUT is used to send data to a server to create/update a resource.The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.

DELETE

Removes all current representations of the target resource given by a URI.

Postman

Postman is an interactive and automatic tool for verifying the APIs of your project.

It is a Google Chrome app for interacting with HTTP APIs and presents a friendly GUI for constructing requests and reading responses.

It works on the backend, and makes sure that each API is working as intended.

Thank You.