MONASH University

FIT5032 - Internet Applications Development

Introduction to Web Development & ASP.NET

Prepared by - Jian Liew

Updated by ABM Russel

Software Stack for Web Development

- Before we move into talking about web development, there is a need to understand the difference software stacks.
- A software stack is a set of software subsystems or components needed to create a complete platform.
- Linux, Apache, MySQL, PHP used to be one of more the popular software stack in the yesteryears. This is known as the **LAMP** stack. (Majority of companies still run this)
- MongoDB, Express, Angular, NodeJS is one of the more popular one in the recent years. This is known as the **MEAN** stack. There is also **MERN**, which replaces angular with React. We will show you this later in the semester. (This is now the "it" thing)
- There is also of course the Java (Enterprise Edition) EE stack. Java EE is called JakartaEE.
- For this subject, we will be using the **Microsoft Technology** software stack. This includes, the .NET Framework, with IIS and MS (Microsoft) SQL Server.
- Materials in this unit will be based on the Microsoft Software Stack.

Different Software Stacks

We will be using this. ⬇

LAMP Stack	Java EE/Jakarta EE Stack	MEAN Stack	Microsoft Stack
Linux	JSF or PrimeFaces	AngularJS	Frontend Technology (Razor)
PHP	JavaEE	Express	.NET Framework .NET Core
Apache	Tomcat, Apache, GlassFish or Payara	Node JS	Internet Information Services (IIS)
MySQL	Derby or PostgreSQL	MongoDB	Microsoft SQL Server

Understanding The Layers

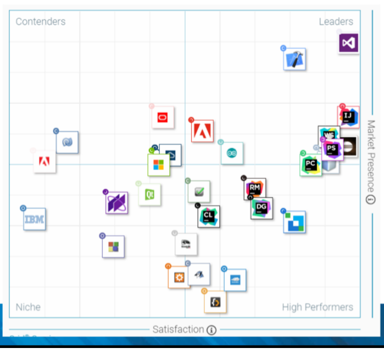
The idea of a web development software stack is to know the various technologies that are involved. For example, the Microsoft stack involves the use of

- Front End Framework or Technology (Razor)
- .NET Framework (This will be the server side programming technology)
- IIS (This is the server itself)
- MS SQL Server (This is our database technology, if you have done FIT9132 - Introduction to Database, you would know that it uses Oracle DB. This here, is Microsoft SQL)

In our Microsoft stack, there is a need to understand that each layer **can be potentially interchanged**. For example, if you choose not to use MS SQL, you can potentially replace this database with PostgreSQL for example. PostgreSQL is another example of an enterprise level database. However, normally this is should not be done as Microsoft has its own ecosystem of technologies.

Microsoft Visual Studio

- Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft.
- It is used to develop computer programs, as well as web sites, web apps, web services and mobile apps.
- Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.
- When we are using a Microsoft Technology Stack (.NET Framework, IIS, MS SQL), we will almost always use Microsoft Visual Studio as our IDE of choice.
- You will learn more about Microsoft Visual Studio in the labs.



ASP.NET Overview

ASP.NET is part of the .NET Framework.

ASP.NET offers three different frameworks for creating web applications

- Web Forms
- MVC
- Web Pages

Each of these frameworks targets a different development style.

The selection of this depends on your programming assets (knowledge, skills and development experience)

For this subject our focus will be using the MVC Framework.

Overview of the different frameworks in ASP.NET

Framework	Development Style	Expertise
Web Forms	Rapid Development using a rich library of controls that encapsulate HTML markup.	Mid-Level Advance RAD
MVC	Full control over HTML markup, code and markup separated, and easy to write test cases.	Mid-Level, Advance
Web Pages	HTML Markup and your code together in the same file.	New, Mid-Level

We will be using this.

What is RAD?

- Rapid application development (RAD) describes a method of software development which heavily emphasizes rapid prototyping and iterative delivery.
- The RAD model is, therefore, a sharp alternative to the typical waterfall development model, which often focuses largely on planning and sequential design practices.
- Rapid application development has become one of the most popular and powerful development methods, which falls under the parental category of agile development techniques.
- Keep in mind that, whenever someone uses the word "agile", it actually means a wide variety of methodologies which might include extreme programming, SCRUM, pair programming and etc. **Kanban which is sometimes used is considered not to be a software development methodology by a variety of people.**

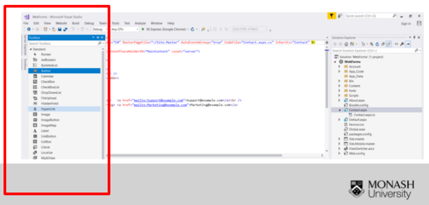
ASP.NET Web Forms

- With ASP.NET Web Forms, you can build dynamic websites using a familiar drag-and-drop, event-driven model.
- A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

The Drag & Drop Feature of ASP.NET Web Forms.

We won't be using this though!

"This is too easy" said Anon, 2016

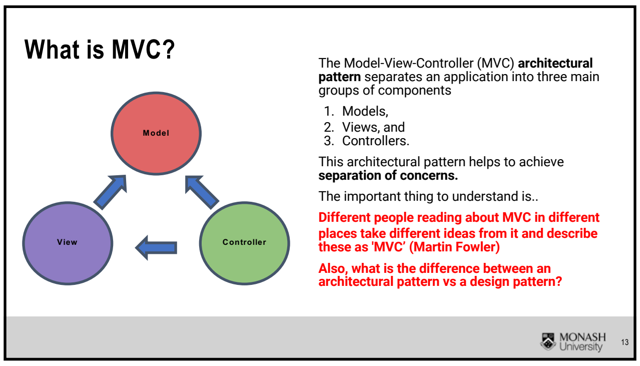


Advantages of .NET Web Forms

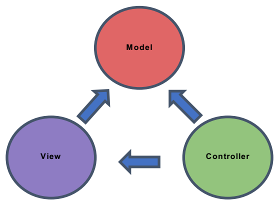
- RAD – Rapid Application Development
  - The mechanisms behind how controls and pages work within an event-driven structure are **abstracted out** so that the developer generally does not need to know the inner workings of the process.
  - **Drag & Drop controls provide** most of the functionality right out of the box. 3rd party solutions are readily available, and control like Grids generate the HTML and JavaScript for the developer.
  - **Applications, complete with validation,** can be quickly developed by simply setting properties on controls.
- Low level of difficulty (Lower learning curve) and considered to be "mature". Majority of companies are now moving away from this though.

Disadvantages of .NET Web Forms

- **Very little control over HTML produced on the page**
  - The ability to simply drag & drop controls enables rapid application development (RAD), but the drawback is that the developer has very little control over what HTML is actually produced on the page. This is actually a major issue for modern web development as it will not allow developers to easily create "responsive" websites.
- The Lifecycle of Web Form is not easily understood by the developer. Because of this it is harder to use 3rd party features in a Web Forms application.



What is MVC?



The Model-View-Controller (MVC) **architectural pattern** separates an application into three main groups of components

1. Models,
2. Views, and
3. Controllers.

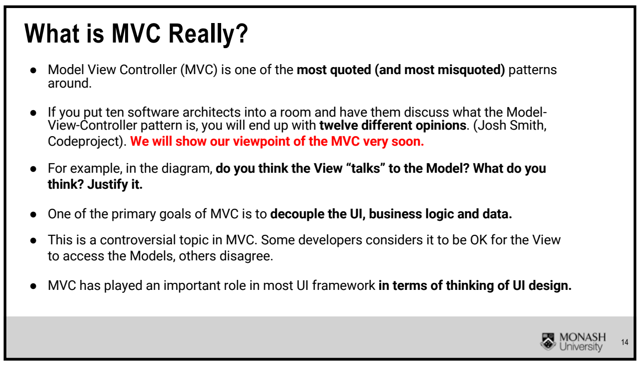
This architectural pattern helps to achieve **separation of concerns.**

The important thing to understand is..

Different people reading about MVC in different places take different ideas from it and describe these as 'MVC' (Martin Fowler)

Also, what is the difference between an architectural pattern vs a design pattern?

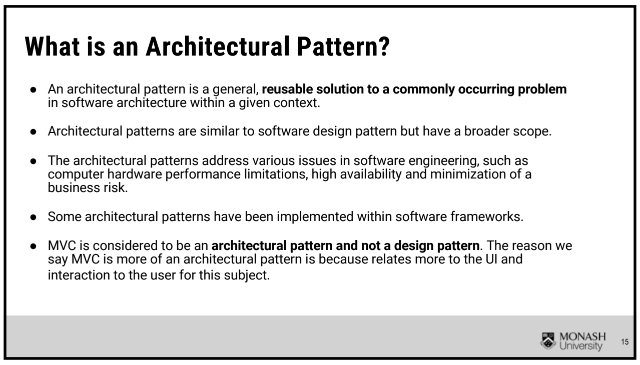
MONASH University 13



What is MVC Really?

- Model View Controller (MVC) is one of the **most quoted (and most misquoted)** patterns around.
- If you put ten software architects into a room and have them discuss what the Model-View-Controller pattern is, you will end up with **twelve different opinions**. (Josh Smith, Codeproject). **We will show our viewpoint of the MVC very soon.**
- For example, in the diagram, **do you think the View "talks" to the Model? What do you think? Justify it.**
- One of the primary goals of MVC is to **decouple the UI, business logic and data.**
- This is a controversial topic in MVC. Some developers considers it to be OK for the View to access the Models, others disagree.
- MVC has played an important role in most UI framework **in terms of thinking of UI design.**

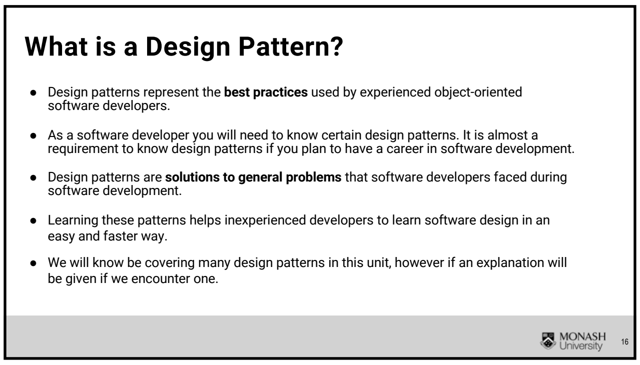
MONASH University 14



What is an Architectural Pattern?

- An architectural pattern is a general, **reusable solution to a commonly occurring problem** in software architecture within a given context.
- Architectural patterns are similar to software design pattern but have a broader scope.
- The architectural patterns address various issues in software engineering, such as computer hardware performance limitations, high availability and minimization of a business risk.
- Some architectural patterns have been implemented within software frameworks.
- MVC is considered to be an **architectural pattern and not a design pattern**. The reason we say MVC is more of an architectural pattern is because relates more to the UI and interaction to the user for this subject.

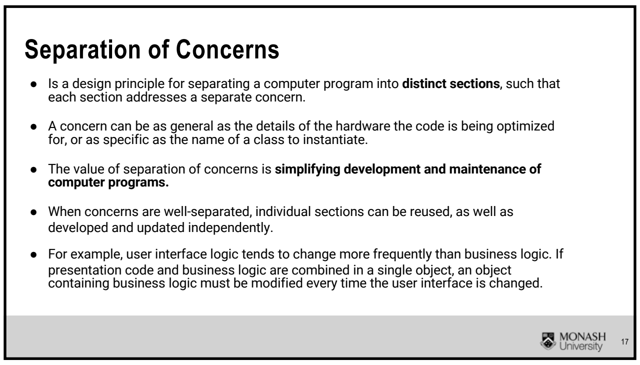
MONASH University 15



What is a Design Pattern?

- Design patterns represent the **best practices** used by experienced object-oriented software developers.
- As a software developer you will need to know certain design patterns. It is almost a requirement to know design patterns if you plan to have a career in software development.
- Design patterns are **solutions to general problems** that software developers faced during software development.
- Learning these patterns helps inexperienced developers to learn software design in an easy and faster way.
- We will know be covering many design patterns in this unit, however if an explanation will be given if we encounter one.

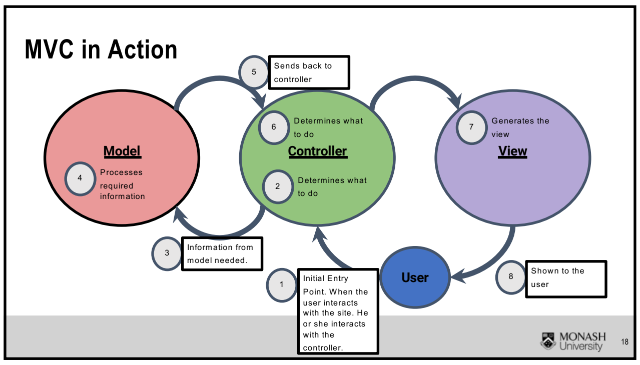
MONASH University 16



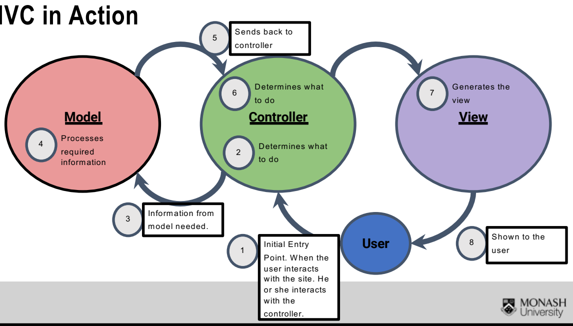
Separation of Concerns

- Is a design principle for separating a computer program into **distinct sections**, such that each section addresses a separate concern.
- A concern can be as general as the details of the hardware the code is being optimized for, or as specific as the name of a class to instantiate.
- The value of separation of concerns is **simplifying development and maintenance of computer programs.**
- When concerns are well-separated, individual sections can be reused, as well as developed and updated independently.
- For example, user interface logic tends to change more frequently than business logic. If presentation code and business logic are combined in a single object, an object containing business logic must be modified every time the user interface is changed.

MONASH University 17



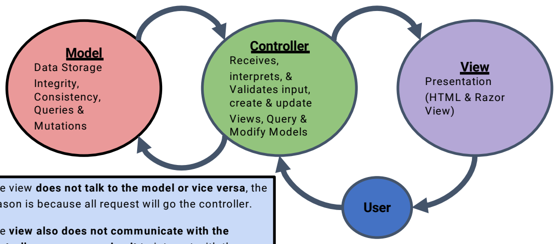
MVC in Action



5 Sends back to controller
Model
4 Processes required information
6 Determines what to do
Controller
2 Determines what to do
7 Generates the view
View
3 Information from model needed.
1 Initial Entry Point. When the user interacts with the site. He or she interacts with the controller.
User
8 Shown to the user

MONASH University 18

ASP.NET MVC


Model
Data Storage
Integrity, Consistency, Queries & Mutations
Controller
Receives, interprets, & Validates input, create & update Views, Query & Modify Models
View
Presentation (HTML & Razor View)
User

The view **does not talk to the model or vice versa**, the reason is because all request will go the controller.

The **view also does not communicate with the controller** as you are **using it** to interact with the controller. **Clicking a button on view would be you interacting with the controller.**

MVC Continued

Model Responsibilities

- The Model in an MVC application represents the state of the application and any business logic or operations that should be performed by it.
- Business logic should be encapsulated in the model, along with any implementation logic for persisting the state of the application.
- Strongly-typed views typically use ViewModel types designed to contain the data to display on that view.
- The controller creates and populates these ViewModel instances from the model.

View Responsibility

- Views are responsible for presenting content through the user interface.
- They use the Razor view engine to embed .NET code in HTML markup.
- There should be minimal logic within views, and any logic in them should relate to presenting content.
- If you find the need to perform a great deal of logic in view files in order to display data from a complex model, consider using a View Component, ViewModel, or view template to simplify the view.

Views in ASP.NET MVC

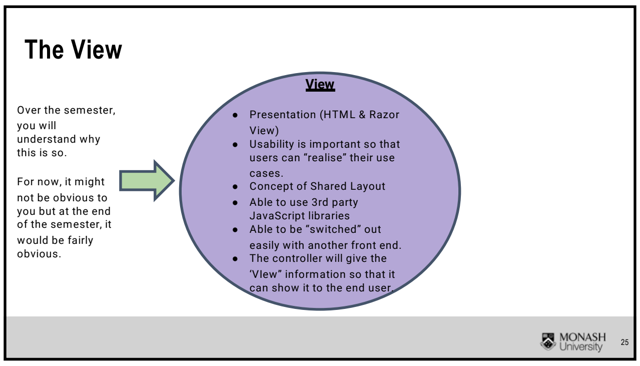
- In the Model-View-Controller (MVC) pattern, the view handles the app's data presentation and user interaction.
- A view is an HTML template with embedded Razor markup.
- Razor markup is code that interacts with HTML markup to produce a webpage that's sent to the client.
- In ASP.NET MVC, views are .cshtml files that use the C# programming language in Razor markup.
- Usually, view files are grouped into folders named for each of the app's controllers.

Benefits of View

- Views help to establish a **Separation of Concerns (SoC)** design within an MVC app by separating the user interface markup from other parts of the application.
- Following SoC design makes your app modular, which provides several benefits:
  - The app is easier to maintain because it's better organized. Views are generally grouped by app feature. This makes it easier to find related views when working on a feature.
  - The parts of the app are loosely coupled. You can build and update the app's views separately from the business logic and data access components. You can modify the views of the app without necessarily having to update other parts of the app.
  - It's easier to test the user interface parts of the app because the views are separate units.
  - Due to better organization, it's less likely that you'll accidently repeat sections of the user interface.

What is Razor?

- Razor is a **markup syntax** for embedding server-based code into web pages. The Razor syntax consists of Razor markup, C#, and HTML. Files containing Razor generally have a .cshtml file extension.
- The default Razor language is HTML. Rendering HTML from Razor markup is no different than rendering HTML from an HTML file. HTML markup in .cshtml Razor files is rendered by the server unchanged.



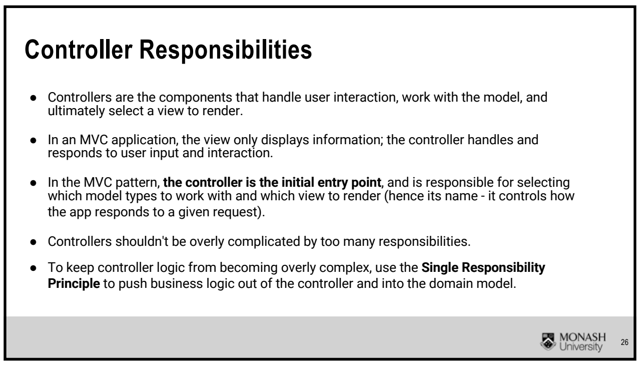
The View

Over the semester, you will understand why this is so.

For now, it might not be obvious to you but at the end of the semester, it would be fairly obvious.

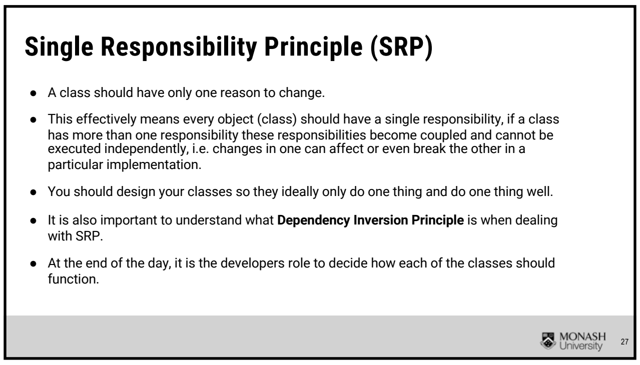
View

- Presentation (HTML & Razor View)
- Usability is important so that users can "realise" their use cases.
- Concept of Shared Layout
- Able to use 3rd party JavaScript libraries
- Able to be "switched" out easily with another front end.
- The controller will give the 'View" information so that it can show it to the end user.



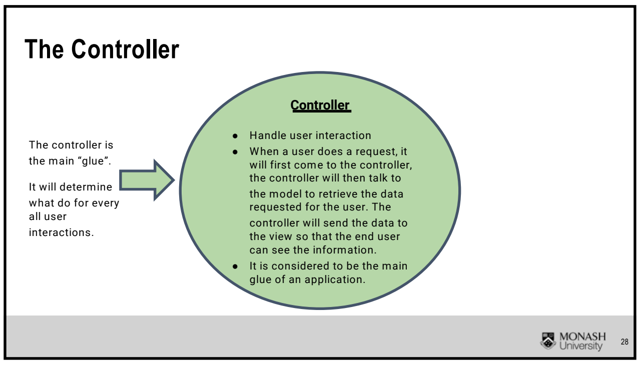
Controller Responsibilities

- Controllers are the components that handle user interaction, work with the model, and ultimately select a view to render.
- In an MVC application, the view only displays information; the controller handles and responds to user input and interaction.
- In the MVC pattern, **the controller is the initial entry point**, and is responsible for selecting which model types to work with and which view to render (hence its name - it controls how the app responds to a given request).
- Controllers shouldn't be overly complicated by too many responsibilities.
- To keep controller logic from becoming overly complex, use the **Single Responsibility Principle** to push business logic out of the controller and into the domain model.



Single Responsibility Principle (SRP)

- A class should have only one reason to change.
- This effectively means every object (class) should have a single responsibility, if a class has more than one responsibility these responsibilities become coupled and cannot be executed independently, i.e. changes in one can affect or even break the other in a particular implementation.
- You should design your classes so they ideally only do one thing and do one thing well.
- It is also important to understand what **Dependency Inversion Principle** is when dealing with SRP.
- At the end of the day, it is the developers role to decide how each of the classes should function.



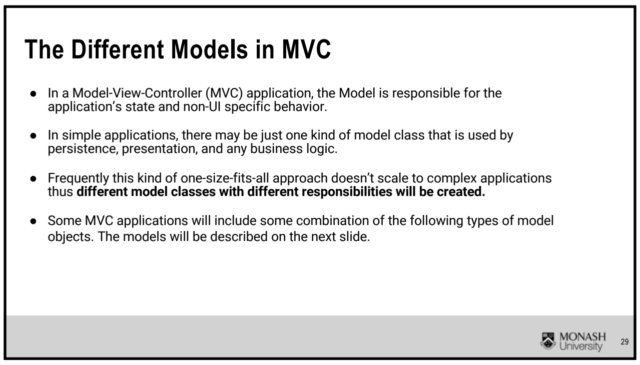
The Controller

The controller is the main "glue".

It will determine what do for every all user interactions.

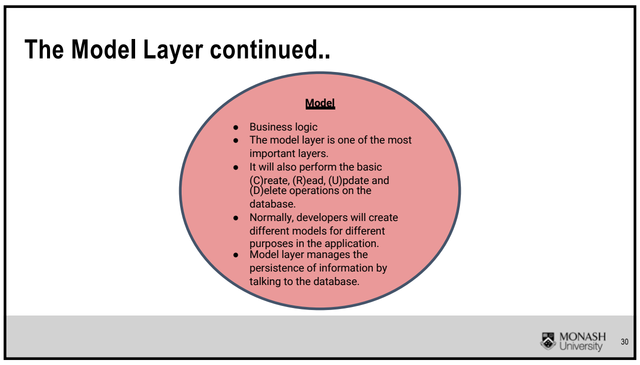
Controller

- Handle user interaction
- When a user does a request, it will first come to the controller, the controller will then talk to the model to retrieve the data requested for the user. The controller will send the data to the view so that the end user can see the information.
- It is considered to be the main glue of an application.



The Different Models in MVC

- In a Model-View-Controller (MVC) application, the Model is responsible for the application's state and non-UI specific behavior.
- In simple applications, there may be just one kind of model class that is used by persistence, presentation, and any business logic.
- Frequently this kind of one-size-fits-all approach doesn't scale to complex applications thus **different model classes with different responsibilities will be created.**
- Some MVC applications will include some combination of the following types of model objects. The models will be described on the next slide.



The Model Layer continued..

Model

- Business logic
- The model layer is one of the most important layers.
- It will also perform the basic (C)reate, (R)ead, (U)pdate and (D)elete operations on the database.
- Normally, developers will create different models for different purposes in the application.
- Model layer manages the persistence of information by talking to the database.

The Different Types of Model

Domain Model | View Model | Binding Model | API Model | Persistence Model

You will learn more how these models will play a role in the future weeks. At the moment, you probably heard of the word “Domain Model” before and perhaps “Persistence Model”

Domain Model

- Many developers choose to encapsulate complex business logic within a domain model. If you have done the subject **System Analysis and Design or Software Engineering** you would have a basic understanding of what a Domain Model is.
- Domain Modeling is a way to **describe and model real world entities and the relationships between them**, which collectively describe the problem domain space.
- The domain model will often include abstractions and services that allow the Controller to operate at a higher level of abstraction.
- The domain model will usually include interface definitions (for services, repositories, etc.) used by the app, as well as persistence-ignorant entities (and some services) that represent the state and behavior of the app’s business logic.
- **It is quite common practice to wrap a service layer over the domain model.**

View Model

- This is not the same as the “View” in the MVC architecture. This View is ViewModel
- In an MVC web application, a ViewModel is a type that includes just the data a View requires for display (and perhaps sending back to the server).
- ViewModel types can also simplify model binding in ASP.NET MVC.
- ViewModel types are generally just data containers; any logic they may have should be specific to helping the View render data.
- There may be many similar ViewModel types, each tailored to the needs of a particular View.

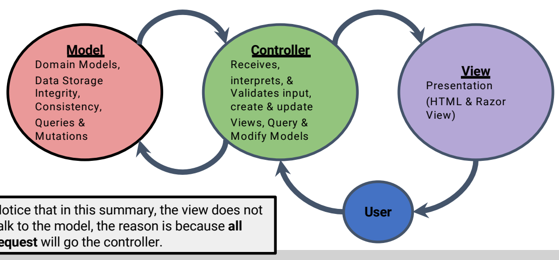
API Model

- If your application exposes an API, the format of the data you expose to clients may be separated from your app’s internal domain model by defining custom API model types.
- This allows you to change your internal model types without impacting clients that may be using your exposed APIs.
- Typically these exposed API models will be used by clients both for read and write operations, so these types will act as both ViewModel and BindingModel for APIs.
- The API Model is a very important concept for **modern web development**. This will be shown to you later in the semester.
- The API Model will make use of the Web API feature of ASP.NET

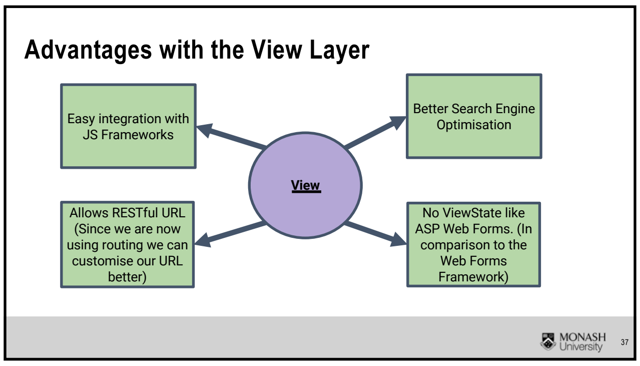
Advantages of MVC

- Enables the full control over the rendered HTML.
- Provides clean separation of concerns(SoC).
- Enables Test Driven Development (TDD).
- Easy integration with JavaScript frameworks.
- Following the design of stateless nature of the web.
- RESTful urls that enables Search Engine Optimisation.
- No ViewState and PostBack events in comparison to ASP.NET Web Forms.

Summary of ASP.NET MVC


Model
Domain Models, Data Storage Integrity, Consistency, Queries & Mutations
Controller
Receives, interprets, & Validates input, create & update Views, Query & Modify Models
View
Presentation (HTML & Razor View)
User

Notice that in this summary, the view does not talk to the model, the reason is because **all request** will go the controller.



Advantages with the View Layer

View

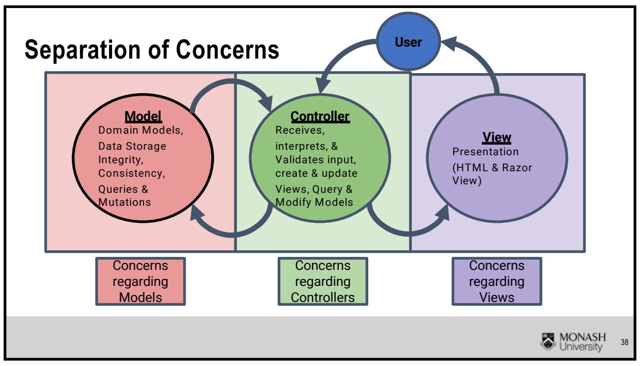
Easy integration with JS Frameworks

Better Search Engine Optimisation

Allows RESTful URL (Since we are now using routing we can customise our URL better)

No ViewState like ASP Web Forms. (In comparison to the Web Forms Framework)

MONASH University 37



Separation of Concerns

User

Model
Domain Models, Data Storage Integrity, Consistency, Queries & Mutations

Controller
Receives, interprets, & Validates input, create & update Views, Query & Modify Models

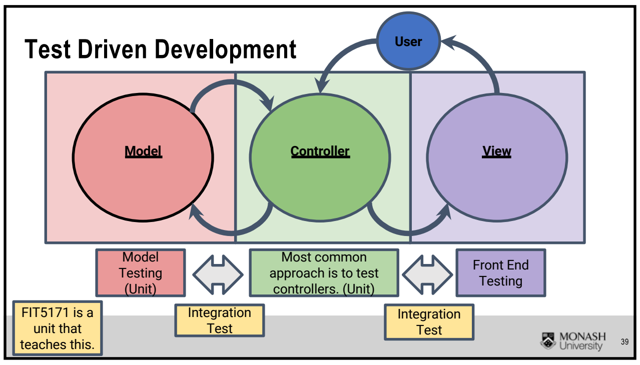
View
Presentation (HTML & Razor View)

Concerns regarding Models

Concerns regarding Controllers

Concerns regarding Views

MONASH University 38



Test Driven Development

User

Model

Controller

View

Model Testing (Unit)

Most common approach is to test controllers. (Unit)

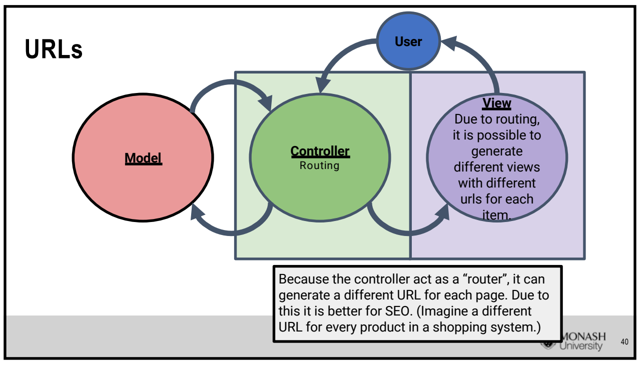
Front End Testing

FIT5171 is a unit that teaches this.

Integration Test

Integration Test

MONASH University 39



URLs

User

Model

Controller
Routing

View
Due to routing, it is possible to generate different views with different urls for each item.

Because the controller act as a "router", it can generate a different URL for each page. Due to this it is better for SEO. (Imagine a different URL for every product in a shopping system.)

MONASH University 40