MVC Framework

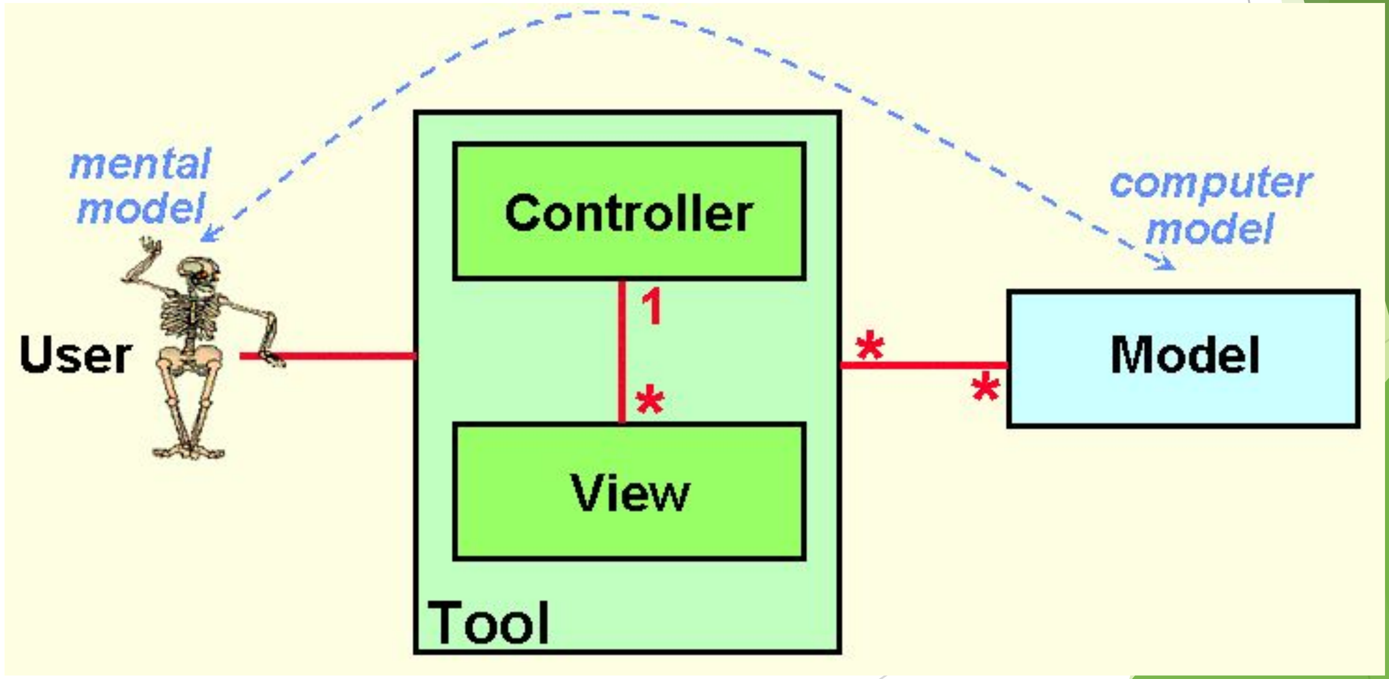
What is MVC?

- Formulated in 1979 by Trygve Reenskaug working at XEROX
- Attempt to solve the problem of bridging the gap between a user's mental model and the digital model that exists on the computer
- In particular, large and complex data set manipulation and control.

Basic Diagram

The following basic MVC (Model View Controller) diagram illustrates two things.

1. The separation of concerns of logic and view
2. The relationship between user and system



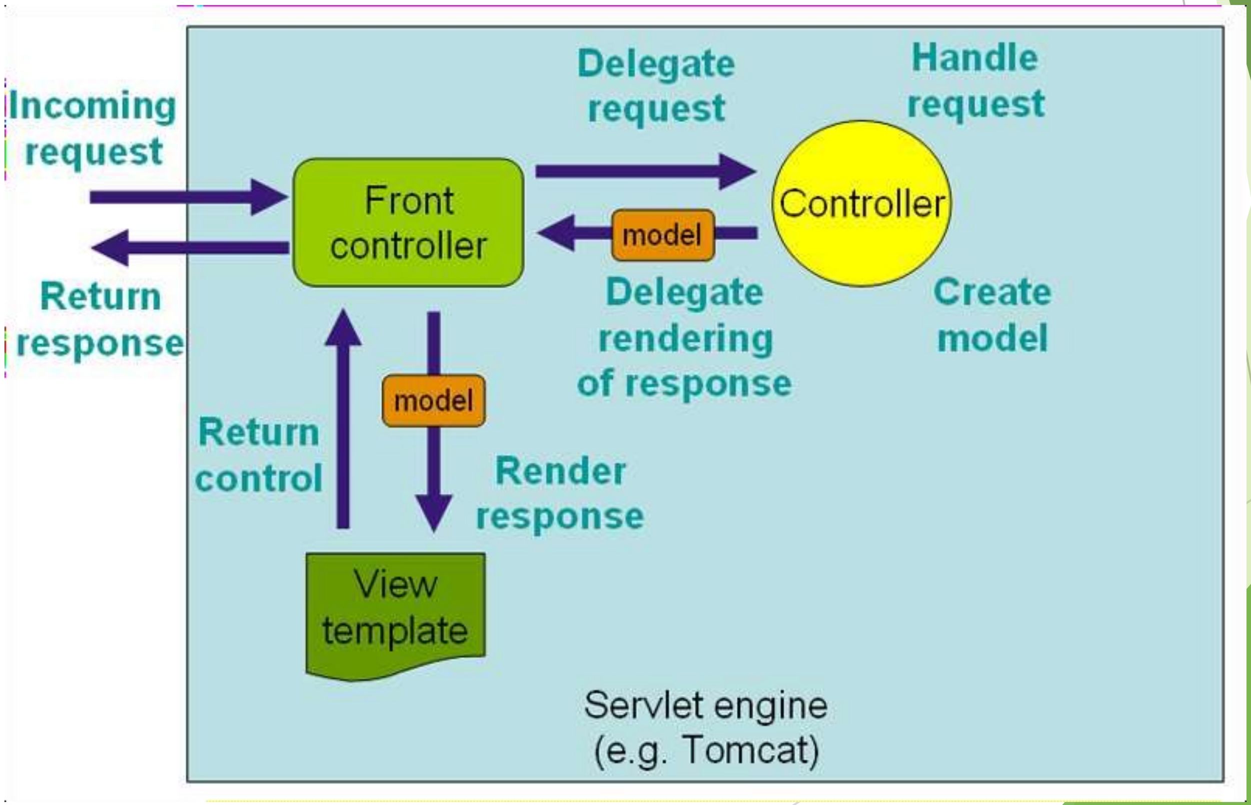
Original Description

Model
The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.
View
The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

- Controller
- Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

Web Diagram

Expanding on the basic model, the web model shows a general process for how the framework reacts to HTTP requests.



Web Description

- **Models:** Classes that represent the problem domain. May contain logic for storing/retrieving from database.
- **Views:** Templates to generate the final view at runtime. Can return CSV, PDF, etc. but typically HTML.
- **Controller:** Responsible for accepting requests from a User and deciding the View to serve up if any.

ASP.NET MVC

- ASP.NET supports three major development models: Web Pages, Web Forms and MVC (Model View Controller).
- ASP.NET MVC framework is a lightweight, highly testable presentation framework that is integrated with the existing ASP.NET features, such as master pages, authentication, etc.
- Within .NET, this framework is defined in the System.Web.Mvc assembly. The latest version of the MVC Framework is 5.0.
- We use Visual Studio to create ASP.NET MVC applications which can be added as a template in Visual Studio

ASP.NET MVC Features

- ASP.NET MVC provides the following features −
- Ideal for developing complex but lightweight applications.
- Provides an extensible and pluggable framework, which can be easily replaced and customized. For example, if you do not wish to use the in-built Razor or ASPX View Engine, then you can use any other third-party view engines or even customize the existing ones.
- Utilizes the component-based design of the application by logically dividing it into Model, View, and Controller components. This enables the developers to manage the complexity of large-scale projects and work on individual components.
- MVC structure enhances the test-driven development and testability of the application, since all the components can be designed interface-based and tested using mock objects. Hence, ASP.NET MVC Framework is ideal for projects with large team of web developers.

- Supports all the existing vast ASP.NET functionalities, such as Authorization and Authentication, Master Pages, Data Binding, User Controls, Memberships, ASP.NET Routing, etc.
- Does not use the concept of View State (which is present in ASP.NET). This helps in building applications, which are lightweight and gives full control to the developers.
- Thus, you can consider MVC Framework as a major framework built on top of ASP.NET providing a large set of added functionality focusing on component-based development and testing.

Before ASP.NET MVC

- ASP.NET Web Forms was the main Microsoft based framework for web.
- Attempted to plant a stateful system into a stateless one.
- Caused tightly coupled integration and minimal extensibility.
- Difficult to understand and maintain.
- No separation of concerns.

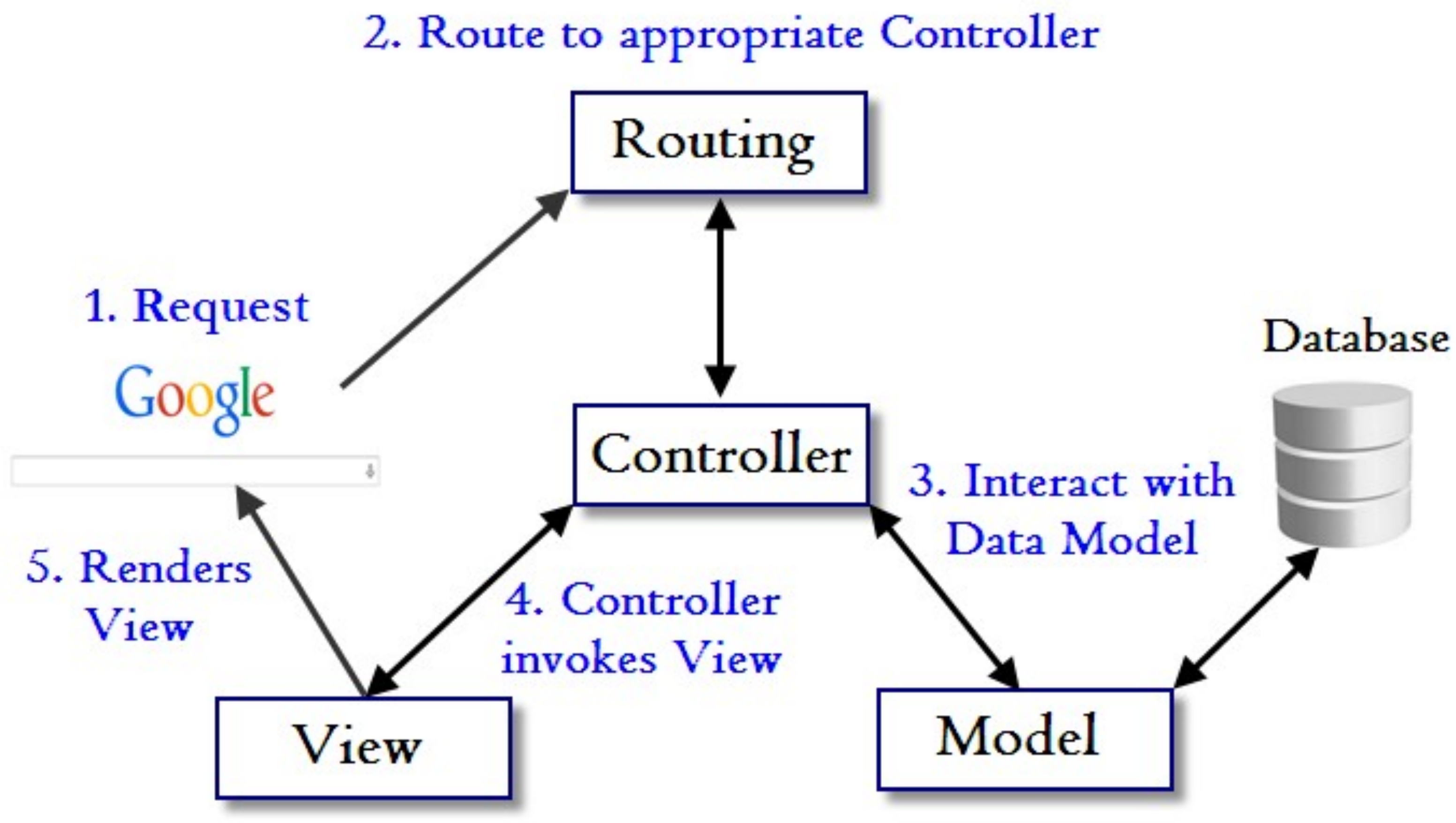
First to MVC

- Ruby on Rails first produced a model view controller framework for web.
- Developers start moving away from ASP.NET to this MVC framework.
- Microsoft, realizing the traction MVC was gaining in the web arena and wanting to keep their technology experts, releases their ASP.NET MVC Framework in 2009.

Why Choose MVC?

- The original MVC framework is a highly adaptable design and has done very well adapting to the web.
- Enhances developer productivity and ease-of-use.
- Features include: HtmlHelpers, Validators, Attribute Based Model Validation, Pluggable Components, Dependency Resolution (Testability)

The MVC Page Lifecycle


2. Route to appropriate Controller
Routing
1. Request
Google
Database
Controller
3. Interact with Data Model
5. Renders View
4. Controller invokes View
View
Model

How does MVC work?

- **Step 1:** Request to ASP.NET stack is handed over to the routing engine.
- **Step 2:** If the Controller is found, it is invoked; otherwise an error occurs.
- **Step 3:** The Controller interacts with the Model as required. If there is incoming data, Model binding is done by ASP.NET MVC to make the incoming data into a strongly typed Model based on the parameters.

How does MVC work?

- **Step 4:** The model if invoked, retrieves or saves appropriate data and returns to the Controller.
- **Step 5:** The Controller then builds a View. MVC Cycles through all View Engines to find a match and renders that view. The ViewEngine returns the ActionResult to the Controller. The Controller uses the ActionResult as a part of its HTTP response.

Model/View Relationship

- In ASP.NET MVC, the Model is a C# class with a set of public properties that can be tagged with MVC specific attributes (like validation).
- Each View has an associated Model that is used to populate and render HTML and is sent back to the server, perhaps modified, on a form (or HTTP) post to the same Action.

Model/View Relationship

- The Controller controls what data is retrieved from the data store, how that data gets fitted to the Model, and which View template gets used to produce the response to the request.
- A single View is associated with a single Action and in ASP.NET must follow the directory structure necessary for the MVC framework.

Web Forms Differences

- In ASP.NET Web Forms, ASP controls were used to populate client side events that would “POST back” to the server.
- MVC does not render events and calls HTTP GET and POST exclusively.
- JavaScript can be used to call additional MVC Controller/Actions while on one page.   This is useful for downloading files or getting JSON data for JavaScript

MVC Benefits

- Ability to interact the same Model in multiple Views.
- Allow the user to customize views for their particular use of the same data.
  - For example, an analyst may wish to see data in a spread sheet while a manager is more inclined to a chart; both people are able to have their View built without changing any business models or recreating logic.

MVC Pitfalls

- Every tool isn't without its faults.
- MVC is more complex than Web Forms due to indirection.
- The event-driven nature has *potential* to make debugging tougher.
- Frequent updates to a View can break the UI. If updates are rapid-fire, batching model updates to send to the view is one possible solution.

MVC – What's Next?

- Multiple MVC Frameworks
- Derivatives of MVC Include:
  - Model View ViewModel (MVVM)
    - Specifically termed by Microsoft
    - Windows Presentation Foundation
    - Adopted by AngularJS
  - Model View Binder (MVB)
    - MVVM Frameworks outside of Microsoft
    - Backbone Framework
  - Model View Whatever (MVW)