

Model View Controller { ASP.NET

By Scott Crooks
& Maggie Wettergreen

Overview {

MVC{

Concept and Origin;
Execution Process;
Popular Frameworks;

};

ASP.NET MVC{

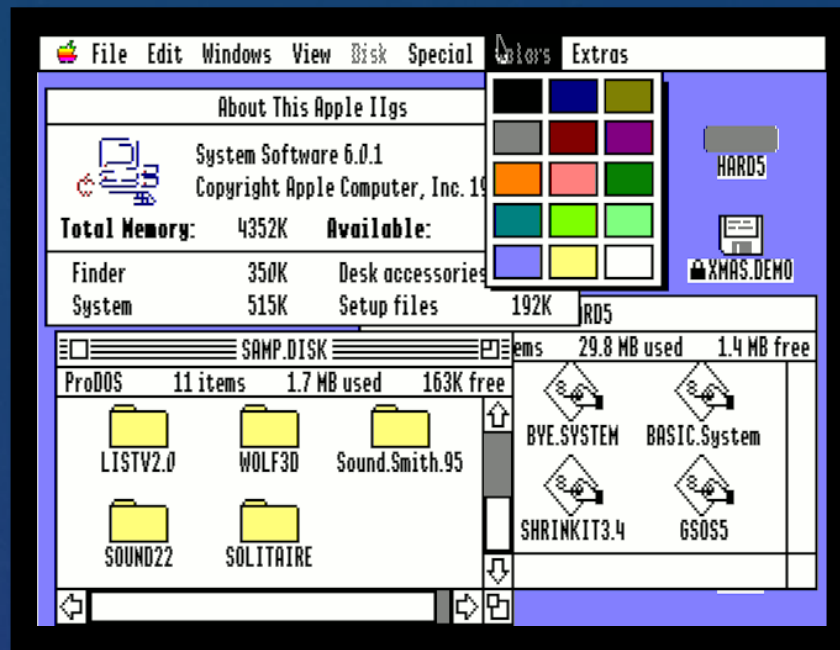
MVC vs. Web Forms;
Features;
Music Store;

};

};



Fist Prototype of a Computer Mouse



Early Apple GUI

Introduction of graphic
“views” in computing

{ 1979

Formulated by Norwegian computer scientist Trygve Reenskaug for Graphic User Interphase (GUI) software design, the MVC architecture was one of the primary outcomes of GUI development.

Presentation tier

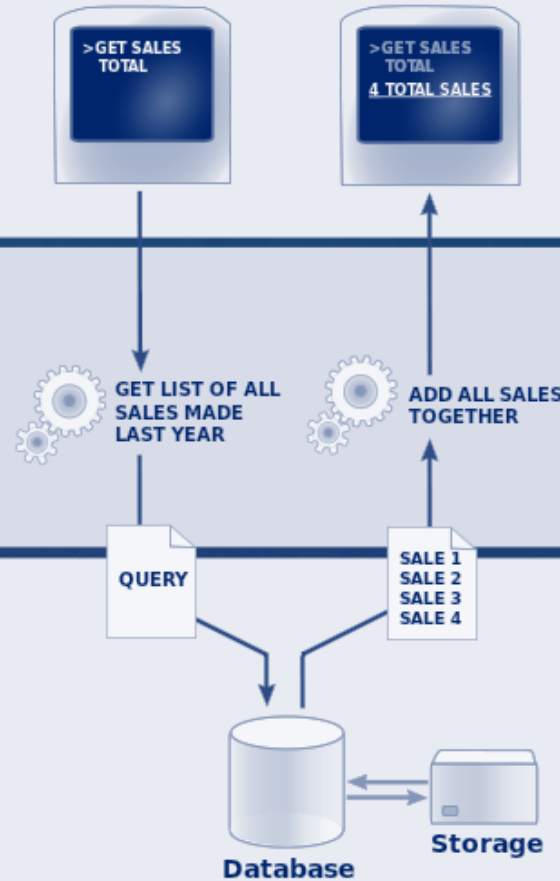
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.



Separates representation of information from user interaction.

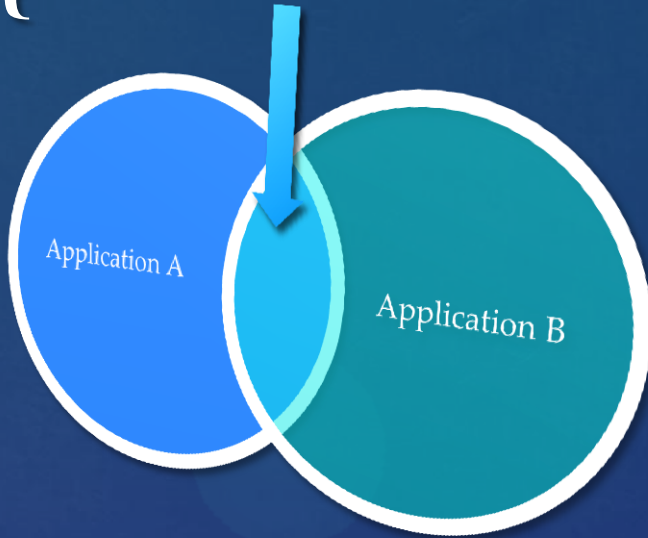
Promotes:

- Code Reusability
- Separation of Concerns

Common 3-Tier Architecture Model

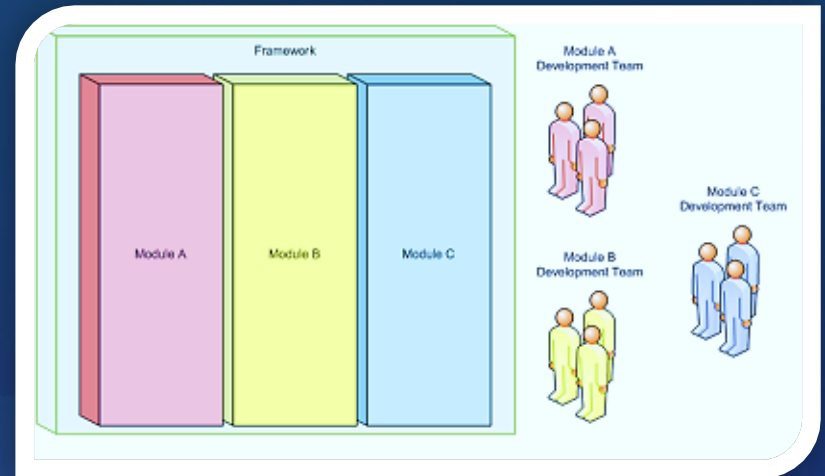
Software Architecture Pattern

{ Code Reusability

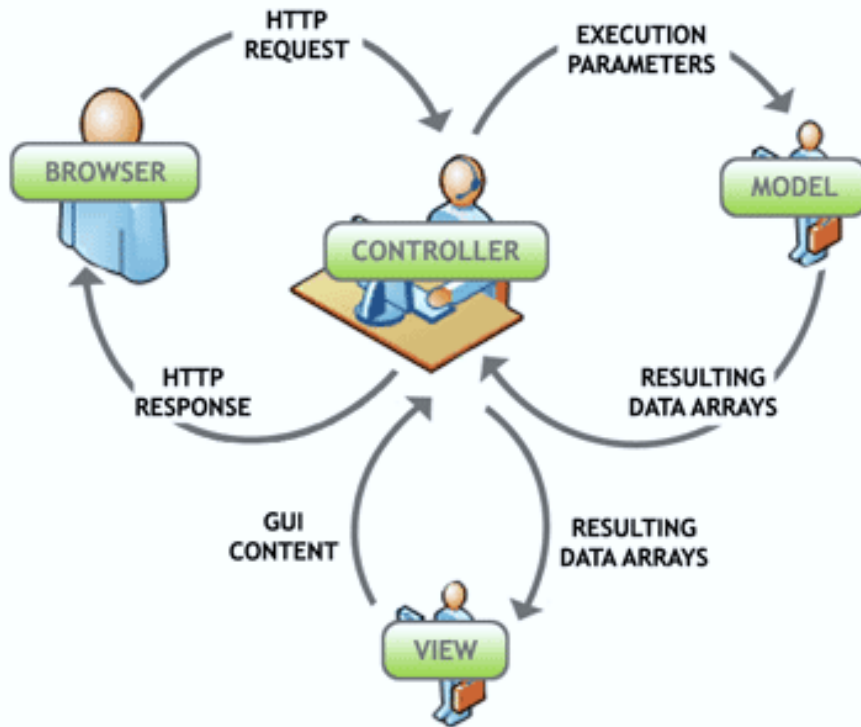


- ⌘ Shortens development
- ⌘ Code Libraries
- ⌘ Design Patterns
- ⌘ Frameworks

{ Separation of Concerns



- ⌘ Improves code clarity and organization
- ⌘ Helps troubleshooting by isolating issues
- ⌘ Allows for multiple teams to develop simultaneously



{ Controller – Mediates input and commands for the model or view

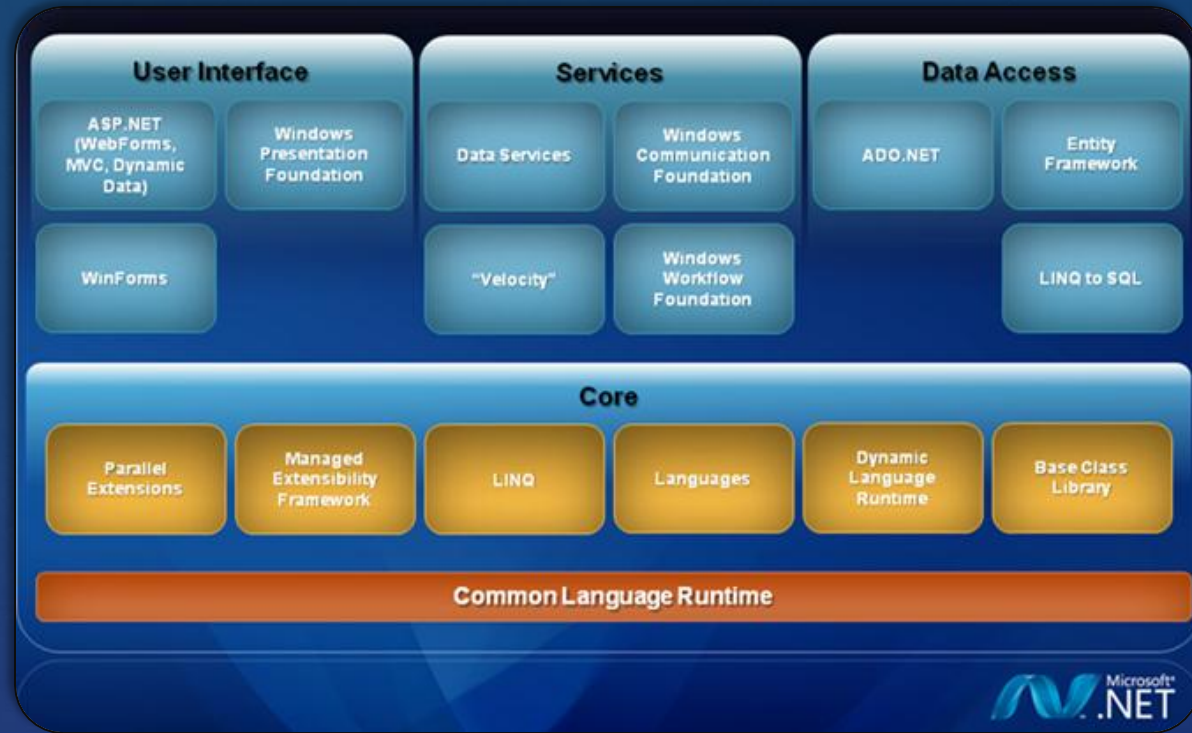
Model – Application data, business rules, logic, and functions.

View – Output and representation of data

Execution Process

Frameworks {

ASP.NET;



ASP.NET 4.0 Framework

PHP (Zend, Symfony, CakePHP, CodeIgniter);

Javascript (Backbone.js, Ember.js, JavascriptMVC);

}

- ⌘ Implements Model-View-Controller Paradigm
- ⌘ Integrates with Existing ASP.NET Features
 - ⌘ Master Pages
 - ⌘ Membership-Based Authentication

ASP.NET MVC

{ MVC

- ⌘ Easier to Manage Complexity
- ⌘ Does not use view state or server based forms
- ⌘ Rich Routing Structure
- ⌘ Support for Test-Driven Development
- ⌘ Supports Large Teams Well

{ WebForms

- ⌘ Preservers State over HTTP
- ⌘ Page Controller Pattern
- ⌘ View state or server based forms
- ⌘ Works well for small teams
- ⌘ Development is less complex

Advantages

- ⌘ Requests routed to Controller::Action
 - ⌘ Action Methods
 - ⌘ Action Results
- ⌘ RESTful

Controllers in ASP.NET MVC

- ⌘ Uses Entity Framework
 - ⌘ Database First
 - ⌘ Model First
 - ⌘ Code First
- ⌘ Database Context
 - ⌘ Describes interactions between entities
- ⌘ Data Annotations
 - ⌘ Describes additional requirements for the model

Models

& Razor

- ⌘ Compact, Expressive, and Fluid
- ⌘ Easy to Learn
- ⌘ Has great Intellisense

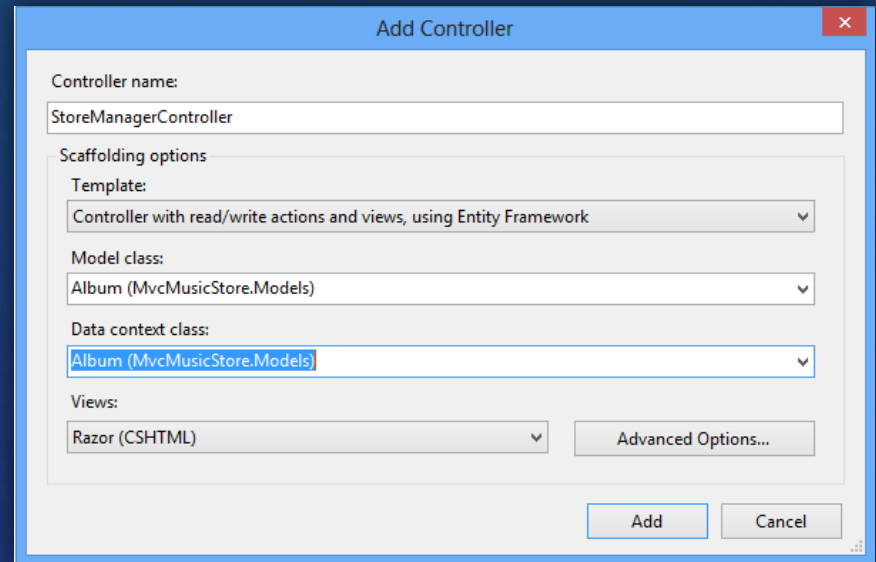
& ASPX Engine

& Dynamic or Strongly Typed

& Partial Views

Views

- ⌘ Scaffolding
- ⌘ Test Driven Development
- ⌘ Internationalization
- ⌘ Many More



Other Features

- ⌘ ASP.NET MVC Implementation
- ⌘ <http://www.asp.net/mvc/tutorials/mvc-music-store>

MVC Music Store