Structure Patterns

by Sophia

WHAT'S COVERED

In this lesson, you will learn what a software structural pattern is and how it is useful in organizing the design of an application or system. Additionally, you will learn about the traditional MVC model as well as the newer SOA structural pattern that has come about as a result of the increase in cloud computing.

Specifically, this lesson will cover the following:

1. Software Structural Pattern

Structural patterns in software development serve as an architectural guide as to where and how different components will be built and how they will interact. Without structural patterns, developers may end up with an unorganized, sometimes **monolithic**, code set that is difficult to manage, fix, and change. Utilizing a structural pattern will make a difference in how easy or difficult the code becomes to maintain by providing organization and strategy for how the software components are organized and how they operate. While there are a lot of patterns available for specific use cases, such as a bridge pattern, a composite pattern, and adapter patterns for smaller projects, we will primarily be focusing on the two most universal patterns used for large-scale or initial build projects.

TERM TO KNOW

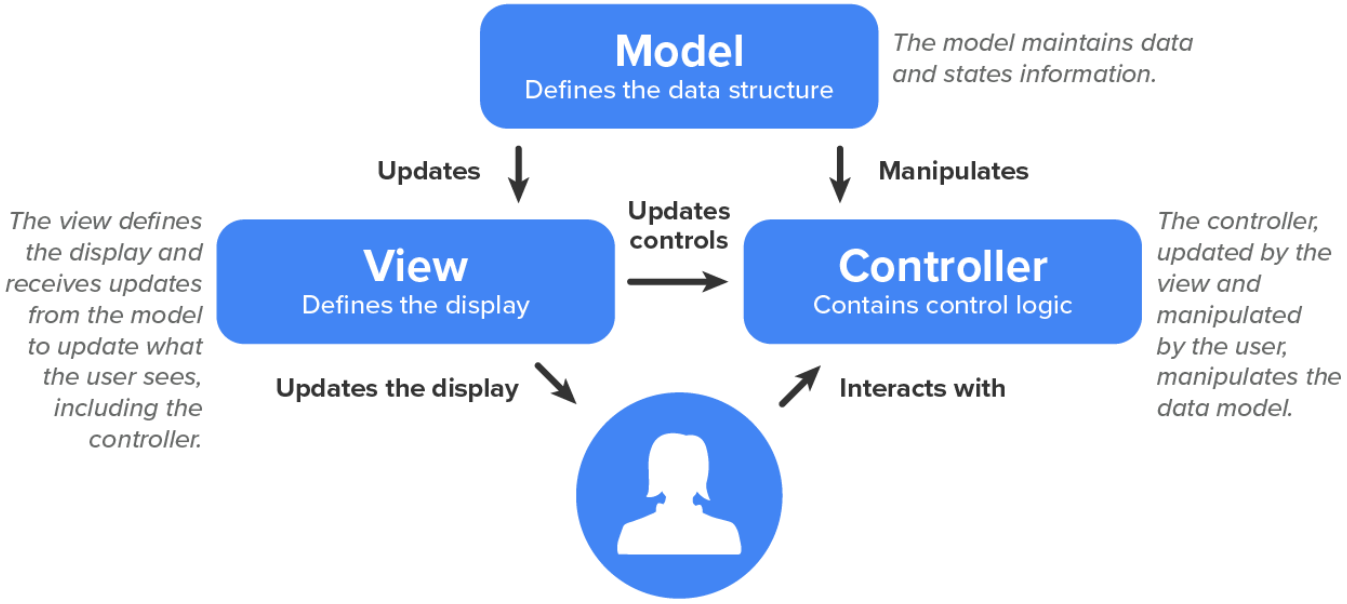
Monolithic
A poor practice of coding wherein all code is combined into a single code structure with no modularization or abstraction.

1a. Model View Controller (MVC)

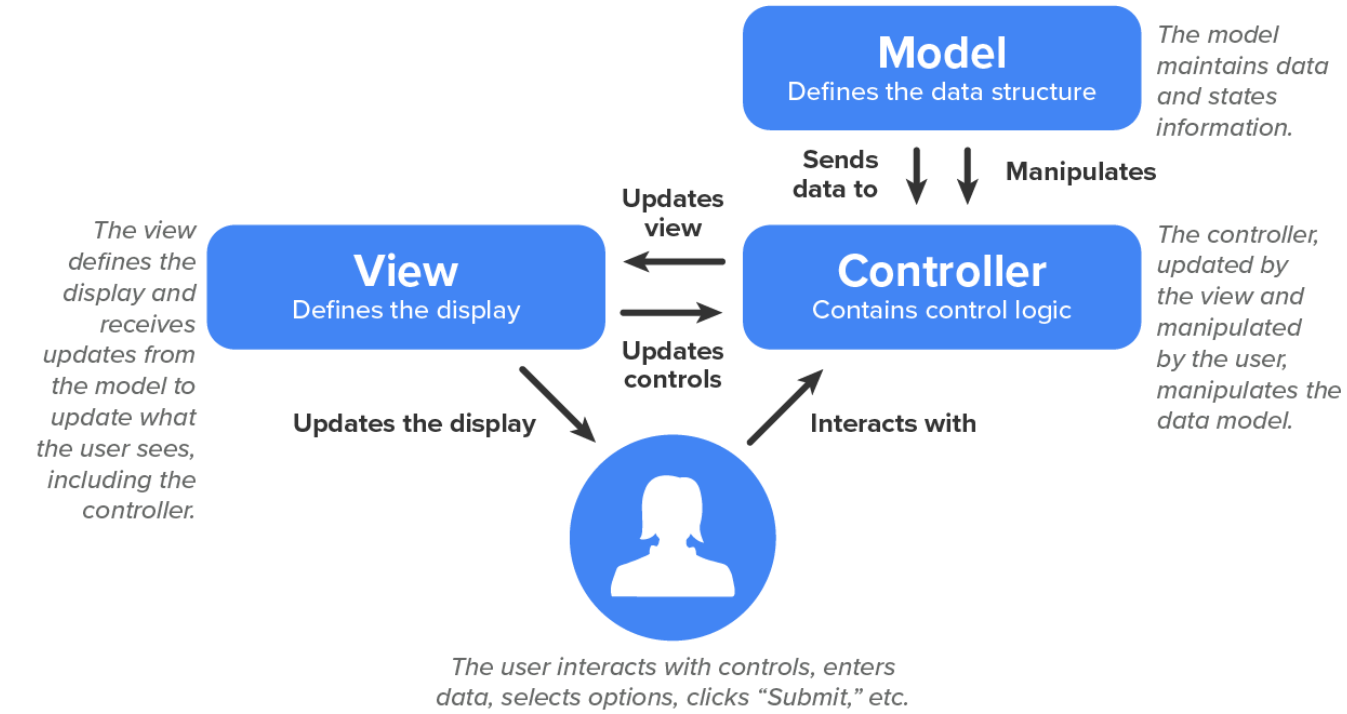
The **Model View Controller (MVC)** pattern, originally developed in the 1970s, is designed to help organize and design any software application in which the user can interact with a complex data structure. This pattern

includes three components: the model, the view, and the controller. The idea is that all functions and operations related to these three components are organized as such and work independently of each other.

MVC Model With Coupled Controller



MVC Model With Decoupled Controller



The model is the data source, structure, and logic that manages the dynamic data set for the application. All of its inner workings should be focused on maintaining the data, organizing the data, and serving the data upon request. The model can also be thought of as an API with **endpoints** that, when requested, perform specific operations on the data, can manipulate the data, and then return the data to the requesting application.

The view is what the user sees, and it is updated with data and information that is requested by the controller. Most often, the view's functions are called when data is received from the model; the data is then passed into those functions, and the functions format and render the data to the screen. The view's functions should only be focused on formatting and rendering the data, which allows a program to be redesigned for other uses, but the same underlying controller and model should be kept.

The controller is the part of the user interface that provides the user with options and form controls. When the user changes the settings in the controller, the controller initiates requests to the model based on the values in the form controls. Once the data is returned by the model, the controller often acts as the mediator between the model and the view by receiving the data from the model and then passing it to the view via standardized function calls to the view.

The decision to use a coupled or decoupled controller model is based on whether or not the model and controllers may be used in other implementations or other views. When the controller handles the passing of data from the model to the view, the controller can use standardized calls to any view to pass the data. The view can then be set up to render the data however it is needed based on what call was made. This effectively decouples the view from the entire structure, allowing it to be rebuilt or replaced. When the view receives the data directly from the model, the view has to have more robust code and functions and needs to anticipate what data it may be receiving. Furthermore, the model has to be programmed to make the correct asynchronous calls to the view. This results in a view component that is more integrated with the entire architecture and thus more difficult to update or replace.

The MVC structural pattern has been an effective model for designing and developing complex applications and systems that interact with complex data structures. However, with so much software, services, and technology going into the cloud these days, a new structural pattern has emerged that takes advantage of APIs and **web services** and only consumes those that they need.

Model View Controller (MVC)
A software structural pattern designed to help organize and design any software application that requires the user to interact with a complex data structure.

Endpoints
The URLs of an API that trigger specific operations by the API.

Web Services
Software that supports the interoperability of different machines or systems through a network.

1b. Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) is a modular approach to developing software applications that operate on the web. The basic idea of SOA is that of an API. Each endpoint of the API provides different functions and operations. SOA is used in web services and **cloud computing** as a method of providing tools, actions, and information to other systems so as to allow larger and/or multiple systems to integrate into them. Each item is a **microservice** in an SOA approach that is oblivious to the actions of other services. It only knows how to request information or actions contained in other services as they are needed. APIs are developed by an organization to provide an array of endpoints that serve as access points to a data source and specific operations.

EXAMPLE

The address *http://edurepository.com/api/users* may be an endpoint that, when requested, triggers a script that collects a list of usernames and IDs from the database and returns it to the requesting system. Similarly, *http://edurepository.com/api/users:4571* may be an endpoint that uses the parameter value 4571 to retrieve information about a specific user.

While SOA and APIs are very similar in how they work and what they do, one key difference is that SOA approaches tend to use a specialized protocol called **SOAP (Simple Object Access Protocol)**. SOAP uses **XML Information Set** as the message structure and transmits across the HTTP (HyperText Markup Language) and sometimes SMTP (Simple Mail Transfer Protocol) application layer protocols. On the other hand, API approaches tend to simply use HTTP requests to specific endpoints. Additionally, SOAP provides developers with features and controls to provide more security and access controls over the SOA.

BIG IDEA

When designing an SOA for an organization or for a specific system, the idea is that you provide a series of operations and actions for web applications through different access points or services. Depending on what the user asks their application to do, it will access the different services as needed in order to fulfill its operation for the user. A theoretical example of this would be writing your own web application that uses the Amazon SOA to retrieve a list of products on Amazon.com based on the parameters selected by the user. The SOA service would receive the SOAP message and may begin accessing other services, as needed, such as an authentication service or a product inventory service to receive a list of products based on the parameters.

While SOA systems may be designed for internal use by the creator's application, they are also often built with a robust set of operations and endpoints that enable third-party entities to develop their own interfaces or applications to make use of the services.

EXAMPLE

SWAPI (Star Wars API) is a public API that is connected to a comprehensive database on the Star Wars fictional universe. The developers first built a database of all information related to the Star Wars canon and then designed the API so the public could make use of the database. If you wanted to design a fun trivia app based on Star Wars, you could design the trivia application to make use of the API endpoints to retrieve data, generate a question, and set the correct answer.

TERMS TO KNOW

Service-Oriented Architecture (SOA)
A method of software development that uses software components called services to create web applications.

Cloud Computing
On-demand high availability system resources for data storage and processing available via the Internet for handling operations without direct actions taken by the user.

Microservice
An architectural pattern in software development wherein the application is composed of multiple, loosely coupled, purpose-specific services.

SOAP (Simple Object Access Protocol)
A messaging protocol specification designed for exchanging information and utilized to design web services.

XML Information Set
XML InfoSet is a W3C specification describing an abstract data model created using XML and used to represent sets of information items.

SUMMARY

In this lesson, you learned about **software structural patterns**. You learned about the components and organization of the **Model View Controller (MVC)** model. You also learned about the newer **Service-Oriented Architecture (SOA)** model and its more modular approach to software architecture.

TERMS TO KNOW

Cloud Computing
On-demand high availability system resources for data storage and processing available via the Internet for handling operations without direct actions taken by the user.

Endpoints
The URLs of an API that trigger specific operations by the API.

Microservice
An architectural pattern in software development wherein the application is composed of multiple, loosely coupled, purpose-specific services.

Model View Controller (MVC)

A software structural pattern designed to help organize and design any software application that requires the user to interact with a complex data structure.

Monolithic

A poor practice of coding wherein all code is combined into a single code structure with no modularization or abstraction.

SOAP (Simple Object Access Protocol)

A messaging protocol specification designed for exchanging information and utilized to design web services.

Service-Oriented Architecture (SOA)

A method of software development that uses software components called services to create web applications.

Web Services

Software that supports the interoperability of different machines or systems through a network.

XML Information Set

XML InfoSet is a W3C specification describing an abstract data model created using XML and used to represent sets of information items.