TASK

Approaches to Web Development

Visit our website

Introduction

WELCOME TO THE APPROACHES TO WEB DEVELOPMENT TASK!

There are many tools, languages and frameworks that can be used for web development today. Deciding on the best approach for web development can be overwhelming. This task will help you to understand some of the most popular approaches to web development and to understand the role of software architecture patterns and how these are used in web development. It will also highlight the strengths and weaknesses of each of these to help you to be better able to make decisions regarding the tools and techniques you will use for various web development projects.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

LANGUAGES AND WEB DEVELOPMENT

There are many different programming languages you could use to create web applications. You can check the **TIOBE index** to see which programming languages are most popular at any given time. The image below shows how languages are ranked on this website.

Mar 2019	Mar 2018	Change	Programming Language	Ratings	Change
1	1		Java	14.880%	-0.06%
2	2		C	13.305%	+0.55%
3	4	▲	Python	8.262%	+2.39%
4	3	▼	C++	8.126%	+1.67%
5	6	▲	Visual Basic .NET	6.429%	+2.34%
6	**5**	▼	**C#**	**3.267%**	**-1.80%**
7	8	▲	JavaScript	2.426%	-1.49%
8	7	▼	PHP	2.420%	-1.59%
9	10	▲	SQL	1.926%	-0.76%
10	14	⇈	Objective-C	1.681%	-0.09%

There is not usually a right or wrong language to use for web development. Each programming language has its own strengths and weaknesses. When deciding which programming language to use when creating web applications, it is good to consider its strengths and weaknesses and choose a language best suited to the project you are working on. A decision will often be made based on personal preference and the preferences and experience of the development team.

FRAMEWORKS FOR WEB DEVELOPMENT

A framework is a basic structure that supports something. A web framework, or a web application framework, is a framework that provides web developers support when developing web applications.

Web frameworks are used by developers to write their code. They are made up of a number of software components that make it quicker and easier for a web developer to create web applications. They are collections of functions, objects, rules and other code constructs designed to solve common problems, speed up development and simplify the different types of tasks faced in a particular domain. Frameworks can be used for both client and server-side code, however the domains, and therefore the frameworks, are very different.

Opinionated and Un-opinionated Frameworks

Web frameworks can either be “opinionated” or “unopinionated”. Opinionated frameworks are those with opinions about the "right way" to handle any particular task. These frameworks often support quick development in a particular domain because the right way to solve a problem is usually well-understood and well-documented. Opinionated frameworks can, however, be less flexible at solving problems outside their main domain. They also tend to offer fewer choices regarding what components and approaches they can use.

Unopinionated frameworks, on the other hand, have fewer restrictions on the best way to glue components together to achieve a goal, or even what components should be used. These frameworks allow developers to use the best tools to complete a particular task but, you need to find these tools yourself. Express is an example of an un-opinionated web framework. There are many other web frameworks. Which framework to choose for web development will depend on various factors including:

- The programming language you will be using. Web frameworks generally support only a few programming languages. You would, therefore, obviously need to look for a framework that you can use to develop web applications in a programming language/languages you can code in.

Programming Language	**Web frameworks**
Python	**Django**, **Flask**
Ruby	Rails, Sinatra
C#	ASP.net
Javascript	Express

- The popularity of the framework. You want to use a framework that is popular because:
    - it is then more likely to be used in industry,
    - it will, therefore, be around for longer and
    - it is more likely to provide better support and documentation for developers.

WEB DEVELOPMENT STACKS

A stack is a term used to describe a collection of technologies that are used together to create a web application. There are a number of web development stacks including:

- The LAMP stack: ie. a stack of technologies (Linux, Apache, MySQL, PHP) that is used to create web applications.
- The MEAN stack: technologies for web development in a MEAN stack are MongoDB, Express, AngularJS, Node.js.
- The MERN stack: as React has become more popular, it has often been used in place of AngularJS in the MEAN stack. This has resulted in the MEAN stack being replaced with the MERN stack.

You will be focussing on the MERN stack for the remainder of this bootcamp.

SPOT CHECK 1

Let's see what you can remember from this section.

1. What are some of the things a web app framework can assist with?
2. When would you use an opinionated framework?
3. When would you use an unopinionated framework?

SOFTWARE ARCHITECTURE PATTERNS

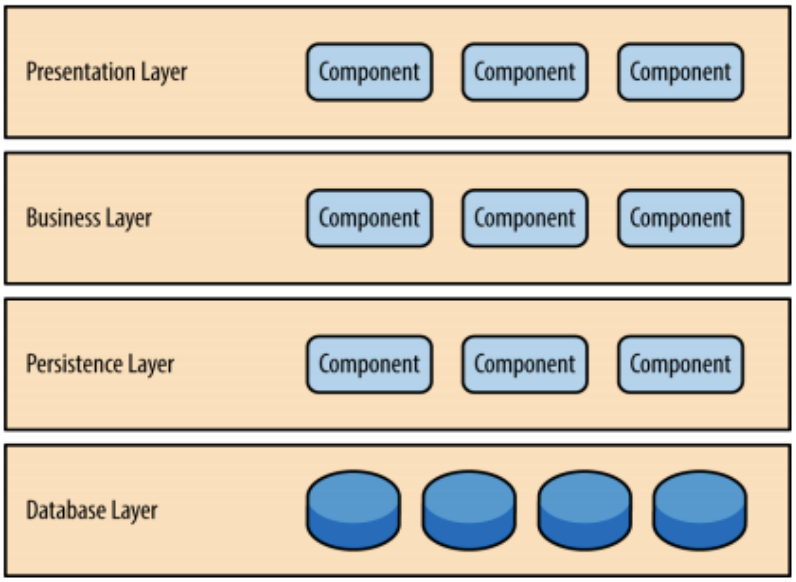
The way we go about web development is not as random as it may at first appear. Most approaches to web development are based on established software architecture patterns. These architecture patterns are well researched and provide ways of thinking about and approaching web development that is known to work.

An architect draws plans that show the basic design of a house: how it will look, what features it will include and, roughly how the plumbing etc. is going to work. Similarly, software architecture patterns provide an overview of the essential characteristics and actions of software applications. Just as not all people would choose to build their house using the same architectural plan, there are different architectural patterns with different strengths and weaknesses which will better suit the design of different software applications. For example, some architectural patterns are more suited to web applications that must be highly scalable while others are more geared for the design of agile apps. This section will describe some common architecture patterns. Software architecture patterns include:

- Layered architecture pattern
- Event-driven architecture pattern
- Microservice architecture pattern

Layered architecture pattern:

This architectural pattern is one of the best known and widely-used architectural patterns. With this architecture pattern, a software application is built using several layers. This is shown in Figure 1 below. This general architecture pattern does not specify how many layers there will be or what each layer will do. It is important though, that with this architecture pattern, each layer is isolated from the other layer in the sense that for the application to work as a whole, each layer need not know how the other layer works. This concept and the reason for it is described in more detail as we discuss the most commonly used type of layered architecture pattern: the model-view-controller (MVC) architecture pattern.



Layer architecture pattern as described by O'Reilly[1]

MVC architecture pattern:

The most commonly used software architecture pattern is a layered architecture pattern, the MVC (model-view-controller) architecture pattern. Before describing the components of the MVC pattern, we will consider the factors that motivate the use of this pattern.

[1] See additional reading for this task

Historically everything needed to make a website work would be contained in one place. For example, all the HTML, CSS and JavaScript for a web page would be in one file. As you no doubt remember from level one of this Bootcamp however, this could cause problems, such as:

- If the user interface and interface logic need to be changed and the business logic and interface logic are coded together, then the code for the entire app will need to be changed.
- Interface logic is usually more affected when an app is used on different devices (e.g. PC vs mobile device) than business logic. If there isn't a clear separation between interface and business logic, the business logic would also have to be rewritten for migration to different devices. This means more testing and increases the likelihood of errors.
- The skills needed for creating attractive interfaces and coding complex business logic are different. Often developers don't have, or are not equally good at, both sets of skills. It would, therefore, be good if different teams of developers could work on interface and business logic separately.

The preceding remarks should make it clear as to why a layered architecture pattern is desirable since each layer can be implemented and updated separately.

The MVC pattern consists of 3 components or layers:

1. The view - this is the component that is responsible for the user interface i.e. it manages how the user views information. With web applications, the view is everything that a user can see in the browser.

2. The model - this is the component that controls the data and business logic of an application. The model often interfaces with a database. It is the heart of the application. The controller and view are dependent on the model but the model is not dependent on the view or controller.

3. The Controller - the controller is the component that controls the interaction between the model and the view. If the user interacts with the view, the controller will know this and can update the model component as needed. The model will perform the necessary business logic and then, if necessary, notify the controller of any changes to the view. The controller should contain orchestration logic, not business logic.

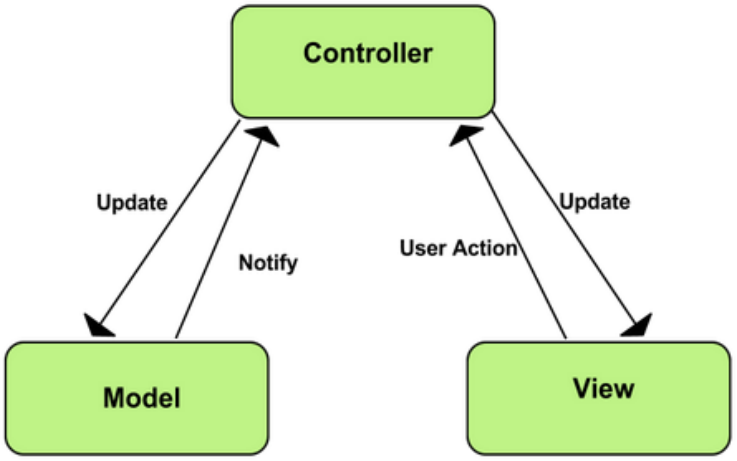


Image source: **https://developer.chrome.com/apps/app_frameworks**

Django's MVT implementation

The MVC pattern has formed the basis of many other patterns. For example, consider how Django implements principles of the MVC pattern using MVT (model, view, template). The components are implemented in Django as described below:

- View - With Django, the view component is more about interpreting what data is presented, as opposed to how the data is presented. The view component in Django, therefore, manages most of the data processing and business logic.

- Template - Django uses templates to describe how the data is presented.

- Model - With Django, the component that interfaces with the database is the model. This is implemented with Django's Object-Relational Mapping (ORM).

The event-driven architecture pattern

This architectural pattern is used to describe distributed asynchronous, highly scalable applications. Pay close attention to this architecture pattern - we will refer back to it in the next task. Understanding this pattern is important in helping us to understand how JavaScript works. The event-driven architecture pattern consists of two main topologies: the mediator and the broker.

Event-driven architecture pattern: the mediator topology

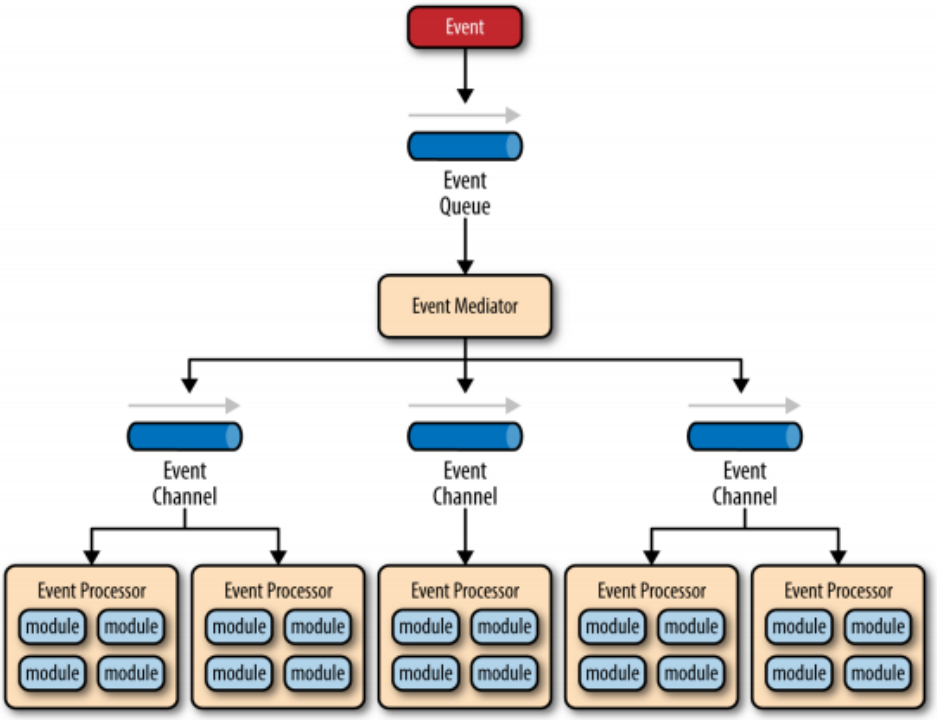
The mediator topology is depicted in Figure 3. The mediator topology is used when a single event needs to be processed using more than one step. As you can see, it is made up of several components including:

- Event queue: as events are triggered they are put onto an event queue where they wait until they can be passed on to the event mediator.

- Event mediator: since the event needs to be handled using various steps, you need an event mediator. The event mediator breaks the original event into a set of processing events that are sent to various event channels. The mediator is responsible for orchestrating the processing events, ie determining the order in which events should be processed, which events can be processed at the same time etc. BPEL is a standard XML-like language that describes the data and steps required for processing an initial event.

- Event channels: event channels receive processing events from the event mediator and pass them on to event processors.

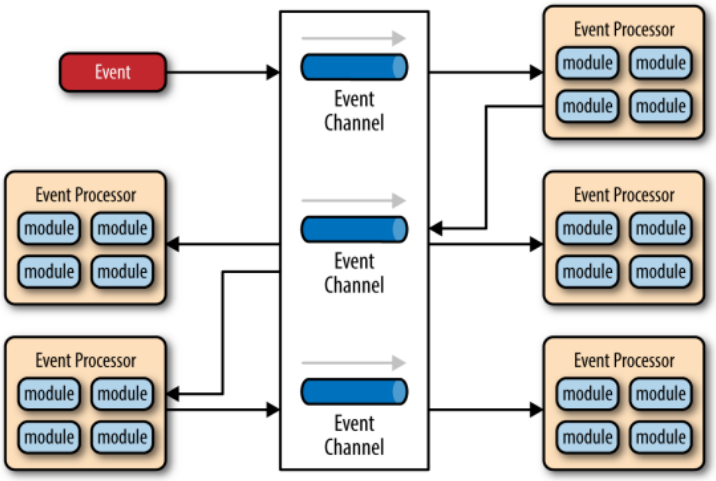
- Event processors: here the actual work happens. The event processor receives the process event and executes the logic to process the event.



Event-driven architecture pattern: the mediator topology as described by O'Reilly

Event-driven architecture pattern: the broker topology

The broker topology is depicted in the next image. As you can see, it works in a similar way to the mediator topology, only there is no mediator. This topology is used when there is relatively simple event processing and an event mediator is not needed to do event orchestration.

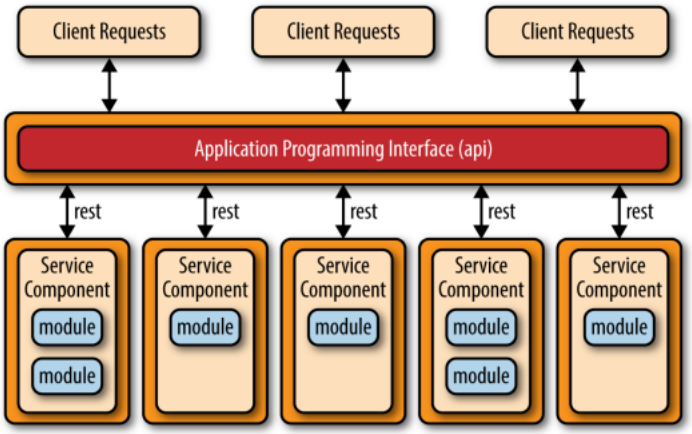


Event-driven architecture pattern: the broker topology as described by O'Reilly

The micro-services architecture pattern

This architecture is used for systems that are made up of decoupled, distributed service components. Service components are units of code that can vary in complexity from a single module of code to an application. Clients can access and use these service components using an interface. The interface layer abstracts the complexity of the service components and exposes to the clients the information they need to use these components.

The concepts described in this topology should sound familiar to you because you have encountered Restful web services before which are based on this architecture pattern. See the task "Modular design for code reuse" in the beginner level of this bootcamp if you need to review this concept.



The micro-services architecture pattern as described by O'Reilly

Compulsory Task 1

Follow these steps:

- Create a text file called **Approaches to web development task.txt**. In this file, answer the following questions:
    - Explain the MVC pattern in your own words.
    - Explain how you think the MVC pattern could apply to the applications you have built with the MERN stack so far.
    - Explain, in your own words, how JavaScript uses the event-driven architecture pattern.
    - On which architectural pattern discussed in this task would you say a Restful API is most based? Justify your answer.

Completed the task(s)?

Ask your mentor to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.

SPOT CHECK 1 ANSWERS

1. They can help solve common problems, speed up development and simplify the different types of tasks faced in a particular domain.
2. Opinionated frameworks are great if you need to develop something quickly in a particular domain because the right way to solve a problem is usually well-understood and well-documented.
3. Unopinionated frameworks are great for projects that need more flexibility because they have fewer restrictions on the best way to glue components together to achieve a goal, or even what components should be used.