# Carson Stevens

# CSCI 445 Web Programming Exam Fall 2020

## (30) Discuss PHP\* web frameworks.

#### NOTE\*: This question is limited to PHP frameworks just because those will relate most closely to what we’re learning this semester. There are certainly excellent frameworks based on a variety of languages.

### a. Why would you use one?

A framework acts as a boilerplate for a project by giving it out of the box functionality. Such frameworks work by extracting common traits and underlying architecture and then bundling them into a reusable template, tools, and a collection of libraries[[1]](#footnote-0). By doing this, developers can focus on the specifics of their project instead of worrying about the nitty-gritty details of the underlying architecture. Because of this, rapid development is often considered a trait of frameworks. This also helps developers stop reproducing code that has already been written; there is no need to reinvent the wheel.

Other perks pertain to the communities surrounding these frameworks. Active communities offer a forum for help and improvement. Such improvements are sometimes integrated into the framework and put into production in the next patch. By doing this, developers will have the collective power of a community to keep code secure and up to date leading to less maintenance in the future. One feature that often leads to features being maintainable is the abstraction of information that changes under the hood, but not for the API calls.



Another reason to use a framework is that you simply have to. Frameworks are used by most major companies[[2]](#footnote-1) which gives them stability and helps the community have trust that these solutions will last. Because of this, as a developer, your team will probably end up interacting with whatever framework your company has chosen. The Mozilla Developer site states, “You don't have to use a server-side web framework, but it is strongly advised — it will make your life a lot easier.”[[3]](#footnote-2)

### b. What problem(s) do they solve?

From the Mozilla Developer Docs[[4]](#footnote-3), they state that web frameworks help solve the problems of working directly with HTTP requests and responses, routing requests to the appropriate handlers, making it easy to access data in the request, abstracting and simplifying database access, and finally, rending data.

When working with requests and responses, a framework helps to abstract the low-level networking primitives into a simplified syntax that allows for high-level code and easier interaction. They also provide an easy way to access the data in those requests such as “GET or POST parameters, cookies, or session data”.

In the same way, frameworks help abstract the complex mapping of resources to “simple mechanisms that map URL patterns to specific handler functions”. One of the biggest perks of using a framework is that they “often provide a database layer that abstracts database read, write, query, and delete operations”. On top of that, they also usually provide a way to help the developer do basic validation on the data.

The last domain mentioned on the Mozilla Docs, rendering data, is crucial and a problem that any modern frameworks must tackle to be successful. In general, these help the developers to “specify the structure of an output document, using placeholders for data that will be added when a page is generated”. This gives developers the ability to create pages that are more dynamic and unique to the user providing an improved experience.

### c. How would you select one? (i.e., what criteria would you consider, what features seem important, etc.)

When choosing a framework, the first thing to note is the external dependencies which result in slower executing times. This complexity and overhead comes, “in the form of classes and libraries loaded before your code even gets called”[[5]](#footnote-4). If bloat ends up causing performance problems, a Domain-Specific Language[[6]](#footnote-5), third-party add-on, or microservice could help alleviate some of the problems, but this aspect should be considered.

Another factor would be the familiarity with the framework and the time it would take to learn to use it effectively[[7]](#footnote-6). Some frameworks abstract processes to higher levels than others which makes them easier to use, but also doesn’t allow developers as much control. It is the strength and weakness of abstraction. Furthermore, some ‘heavier’ frameworks provide more functionality that may not be needed in your project. This would be considered extra bloat that simply slows the execution of your program. Choosing a framework that fits the purpose of the application without extra overhead is optimal.

Two other factors mentioned in the Mozilla Developer Docs were scalability and caching support. When an application has exhausted the benefits gained by caching and has reached the limit of vertical scalability (processing power of hardware), the application will have to scale horizontally across a number of web servers and databases. Once a framework has been chosen, it is not a simple task to switch to a different framework that will work better for your current scale. In that aspect, it is important to consider the scope and overhead that the server may incur while operating. A slow or crashed server doesn’t provide a good user experience. The other option is caching support which becomes useful when the website “can no longer cope with the number of requests it is receiving as user access it”. In premise, caching “stores all or part of a web response so that it does not have to be recalculated on subsequent requests”. While this option works short term, it is just a band-aid for growing overhead.

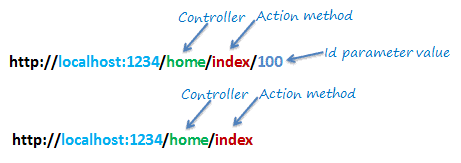
### d. Give a brief explanation of MVC. Also briefly describe routes and controllers. These topics will be much easier to understand when you actually work with a framework.

#### MVC

MVC, or Model-View-Controller, is a template of guidelines used to help a user implement, separate, and connecting the interface, data, and controlling logic. The Mozilla Developer Docs state the purpose of the Model is to “manage data and business logic”. The purpose of the View is to “handle layout and display”. And finally, the Controller “routes commands to the model and view parts”.

State changes within an application are difficult to keep consistent across all scopes of the application. The Model helps us define what data the app should contain. When this data is changed, the model notifies the View so that it can be updated. If more complex logic is needed, the Controller may be notified by the Model as well. When the application receives user input, the Controller will apply whatever logic is intended for the data and then updates the Model and the View. The result will in turn be an updated rendering of the information for the user and an update of the Models data. The Controller also gives developers the ability to only update the Model or just the View depending on the intended application.

#### Routing

Routing in backend terminology “defines the URL pattern and handler information” which is processed to determine the “appropriate handler class or file for an incoming request”.[[8]](#footnote-7) It is the “infrastructure that guides each request to the right controller”[[9]](#footnote-8). This information is parsed from the incoming URL as shown in the Figure. The keep points from routing are that it maps the URL to a file or Controller class, contains URL pattern and handler information (the pattern starts after the domain name), custom routing can be configured and is executed in a hierarchical fashion, and finally, route constraints apply restrictions on the value of parameters (can provide useful information about invalid input). Once the request has been sent to the correct controller, the specified action generates the outputted HTML in the View or modifies the data in Model (or both).

## 2. (20) What is REST? What is CRUD? How are these terms related?

### REST

REST, or REpresentation State Transfer, is “an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other”[[10]](#footnote-9). The first defining factor is the separation of client and server. This factor is important because it allows for the changing of server-side code without affecting the operation on the client-side. The consistency of standards is essential here because as long as both the server and silent know the format of the messages, each can be maintained separately.

The other big component is ‘statelessness’. A REST application is considered stateless if “the server does not need to know anything about what state the client is in and vice versa”.[[11]](#footnote-10) What this boils down to is that the operations on the server’s resources do not depend on the implementation of the user interface. By maintaining this standard, applications “achieve reliability, quick performance, and scalability”.[[12]](#footnote-11) The REST standards also define how the make requests such as the appropriate HTTP verbs (GET, POST, PUT, DELETE), headers and parameters, and of course the path of the request. In response, REST also defines a standard set of response codes for a request (200, 201, 204, etc.). This helps give users and developers consistent feedback about the state of their requests.

### CRUD

CRUD, or CREATE, READ, UPDATE, DELETE, “are the four basic functions that models should be able to do, *at most*”.[[13]](#footnote-12) These operations are performed on the server’s resources. When a model can’t be represented with only these four functions, a separate model should be considered. The purpose of CRUD is simply to provide “a memorable framework for reminding developers on how to construct full, usable models”.

### REST and CRUD Relationship

The acronym for CRUD corresponds to the HTTP verbs mentioned in the REST standards before. CREATE is similar to POST. READ to GET. UPDATE to PUT. And DELETE to DELETE. The CREATE/POST option also us to create a resource in the REST environment and send a body detailing the response and a response code stating how the operating went. The READ/GET method allows us to retrieve information, but never change it (similar to only having read permissions for a file). It also sends a response back with a body and response code. The UPDATE/PUT method pertains to updating the database with the provided information. As per REST standards, it too returns a body and status code. The final method, DELETE removes resources from the server. It is important to note that “calling DELETE on a resource that does not exist should not change the state of the system. The call should return a 404 response code (NOT FOUND) and do nothing”.

Final Notes:

I know the first meme is related to Node.js, but thought it was hilarious. Being invested in Python and using it as a backend previously has probably made me more apprehensive about PHP, so I thought this last meme was really funny. (And who can’t use an extra smile haha.)

1. https://code.tutsplus.com/tutorials/should-you-use-a-php-framework-five-pros-and-cons--cms-28905 [↑](#footnote-ref-0)
2. https://hackr.io/blog/top-10-web-development-frameworks-in-2020 [↑](#footnote-ref-1)
3. https://developer.mozilla.org/en-US/docs/Learn/Server-side/First\_steps/Web\_frameworks [↑](#footnote-ref-2)
4. https://developer.mozilla.org/en-US/docs/Learn/Server-side/First\_steps/Web\_frameworks [↑](#footnote-ref-3)
5. https://developer.mozilla.org/en-US/docs/Learn/Server-side/First\_steps/Web\_frameworks [↑](#footnote-ref-4)
6. https://martinfowler.com/books/dsl.html [↑](#footnote-ref-5)
7. https://code.tutsplus.com/tutorials/should-you-use-a-php-framework-five-pros-and-cons--cms-28905 [↑](#footnote-ref-6)
8. https://www.tutorialsteacher.com/mvc/routing-in-mvc [↑](#footnote-ref-7)
9. https://devnet.kentico.com/articles/routing-and-url-handling-in-mvc [↑](#footnote-ref-8)
10. https://www.codecademy.com/articles/what-is-rest [↑](#footnote-ref-9)
11. https://www.codecademy.com/articles/what-is-rest [↑](#footnote-ref-10)
12. https://www.codecademy.com/articles/what-is-rest [↑](#footnote-ref-11)
13. https://www.codecademy.com/articles/what-is-crud [↑](#footnote-ref-12)