Stephen Cardone

CS-230-J3533

Module 4 Journal

The Client-Server pattern is an excellent framework that allows software engineers to effectively solve problems when developing software. With the client-server pattern, the client is controlling the lifecycle and actions of the program. This is vital because this makes it so that the server does not need to manage the potentially millions of different sessions that are accessing the server. The client can control its own session which splits the load up amongst all the clients that are using that software. With each client controlling its own app flow and lifecycle, how does the server fit in? The server is available to the client to get, and set data. The reason that a client-server pattern is especially good for filling requirements, is because you can make changes at the client level, with no affect on the server. Say you are responsible for the development of a client application. You get through the development and have a working product, but then the requirements change and you need to make some changes to the application. With the Client-Server pattern, you can freely make changes to you application, but the server requests will not change. You might need to move when or where the server requests are on the client side application, but you can easily make changes to fit your requirements with little to no changes to the server.

REST (Representational State Transfer) is a style of communication between client and server. It has proved to be a very effective way for communication over the internet. In the REST system, there are some interface constraints. When these constraints are met, then the client and server are able to share data back and forth which allows for communication. The data being shared in a RESTful system is abstracted into a Resource. A resource can be for example JSON data, which is a text syntax that is easily readable by both humans and computers. In JSON, you can have a key, value pair that maps an attribute to a data point. Assuming that the REST constraints are met, you can create JSON in your application and post to the server, or the Server can store, or generate JSON data and send it to the client.

From the client side, the developers need to be able to identify what they need to do to allow for the software to run on whatever clients are included in their requirements. This would entail picking the correct programming languages that can properly function on the targeted client platforms. In the client side of our game application, in order to add more users, we would need to request that the server add a new user to its known users database. Without this, the person would try to play the game, but they would not be allowed to because they are not known to the server and are therefore unable to sign in. If The Gaming Room asked me to host the application on a fourth and fifth client, I would make sure that the client’s they are requesting support java. If they do, then it should be reasonable for us to include it in the usable clients. There may be some minor updates or changes that would need to be made to support those clients, but it would not be nearly as much work as trying to support a client that doesn’t support java. Overall, the Client-Server pattern is a very effective and easy to use framework that allows for communication and data transfer over the internet.