MK Fowlkes

CS 230

Module 4 Journal

Paul.fowlkes.snhu.edu

The Client-Server architectural pattern is one of the most widely used patterns. This network is made up of a server on the back end that communicates with several clients and supplies a service. The most common example is the World Wide Web page. The World Wide Web. Because the information is stored on the server, multiple clients can access it. Unlike the client machine. Setting up a system in this manner is a huge help to developers. Since most of the computational power is managed on the server. Taking care of the processing power on the server side cuts many of the complications that would come when figuring out, since most of the computational power is managed on the server. Handling processing power on the server side drops many of the complications that would appear when calculating computing power for different platforms. Because the server does the hard lifting, building a Client-Server Pattern allows a web-based game to run on various devices, including mobile ones. Because the Client and Server live in various locations and have different sets of concerns, they can evolve independently.

The Interface facilitates communication between the two, allowing the Server to accomplish its task. When the Client asks information, the Server answers in REST API style, using a URL to identify the data required and then providing the answer via an HTTP request. The normal method for passing data is to deliver it in JSON format. JSON format is a language that both computers and humans can understand. This format enables a multilayered approach to architecture, allowing a developer to hide data behind a security system, most frequently a username/password system. Layering is also an excellent resource for load balancing and performance enhancement.

One of the defining characteristics of REST is that all information is Stateless, which means that the server has no idea what state the client is in. This independence enables both parties to comprehend any messages received. REST also supports caching, which eliminates some interactions and improves scalability and performance.

On the client side, to ensure that all of the clients required to run the website can do so, you must ensure that they can all interact with the server. To avoid timing out, the simplest approach is to add additional server-side capabilities, which would allow the database to grow while still allowing additional clients to access during games. The capacity to receive payments is a critical function that The Gaming Room would require. To earn income, the database will require a safe method of storing payment information. The multitiered REST architecture would include a built-in security system for payment information. If The Gaming Room wanted to add two more clients (such as XBOX and PlayStation), they would need to modify the server code to allow it to communicate with the clients. The benefits of REST would truly shine through in this scenario because the Server code could be modified without affecting existing clients.

Franklin, M. J., Carey, M. J., & Livny, M. (1997). Transactional client-server cache consistency. *ACM Transactions on Database Systems*, *22*(3), 315–363. https://doi.org/10.1145/261124.261125

Ghose, A., & Koliadis, G. (2007). Auditing business process compliance. In *Lecture Notes in Computer Science* (pp. 169–180). https://doi.org/10.1007/978-3-540-74974-5\_14

Hashmi, M., Governatori, G., & Wynn, M. T. (2012). Business process data compliance. In *Springer eBooks* (pp. 32–46). https://doi.org/10.1007/978-3-642-32689-9\_4

Telle, K. (2012). Monitoring and Enforcement of Environmental Regulations: Lessons from a Natural Field Experiment in Norway. *Social Science Research Network*. https://doi.org/10.2139/ssrn.2002986

Yates, A., Beal, K., Keenan, S., McLaren, W., Pignatelli, M., Ritchie, G. R. S., Ruffier, M., Taylor, K., Vullo, A., & Flicek, P. (2014). The Ensembl REST API: Ensembl data for any language. *Bioinformatics*, *31*(1), 143–145. https://doi.org/10.1093/bioinformatics/btu613