Understanding REST

CLIENT-SERVER COMMUNICATION Traditional full-stack applications use REST as their API paradigm; which employs HTTP methods for CRUD operations. Previously we have already been using HTTP methods for CRUD operations – without following clear constraints – among files and user interactions like creating a blog post with server-side languages like PHP.

However, when using a REST API we are using these HTTP methods on RESTful resources. For example, a RESTful resource could be a blog post. A user can read blog posts with a HTTP GET from the application server or create a new blog post with a HTTP POST on the application server.

13 1

A REST API connects client and server applications without them needing to be implemented in the same programming language. They only need to offer a library for sending and receiving HTTP requests and responses. REST is a communication paradigm which is free of a data format (it has been XML in the past, but these days it is JSON) and programming language.

13

A modern alternative to REST is GraphQL for APIs between clients and servers. GraphQL isn't bound to a data format either, and in contrast to REST not bound to HTTP, but most often you will see HTTP and JSON used here as well.

With the technology discussed up to this point, full-stack applications decouple client and server applications. Both communicate via a well-chosen API (e.g. REST or GraphQL). While the client application renders everything necessary for the web application in the browser, the server application handles requests from the client for read and write data.

Exercises: Learn how to create a REST API with Node.js. Read more about why you would want to use GraphQL instead of REST. Learn how to use GraphQL for full-stack JavaScript applications.

Links and Resources

- Architectural Styles and the Design of Network-based Software

Architectures

- Client-Server communication