Roy Fielding, one of the principal authors behind the HTTP specification, wrote a dissertation laying out the theoretical foundation for REST. Based on his work, this article takes a look at the motivation behind REST, as well as its primary fundamental principles.

REST as an Architectural Style for the Modern Web

The famous academic work by Roy Fielding is called “Architectural Styles and the Design of Network-based Software Architectures.” In the [abstract](https://www.ics.uci.edu/~fielding/pubs/dissertation/abstract.htm), Fielding introduced REST as an architectural style for design and development on the modern Web.

The term “**RE**presentational **S**tate **T**ransfer” is derived from the properties of the generic interface used for communication. We’ll be going deeper into those details in a later blog article. For now, we’ll be focusing on the term “architectural style.” Fielding’s definition of an [architectural style](https://www.ics.uci.edu/~fielding/pubs/dissertation/software_arch.htm) is:

*An architectural style is a coordinated set of architectural constraints that restricts the roles/features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style.*

Here, the word “constraints” can also be understood to be “properties.” Thus, REST is an architectural style and describes the properties in the elements of (modern) web architecture.

Hypermedia is the Focus

When it comes to REST, hypermedia is of central importance. In the [very first sentence](https://www.ics.uci.edu/~fielding/pubs/dissertation/abstract.htm) of his dissertation, Fielding describes how the success of the Web can largely be explained by its attribute of being a distributed hypermedia system:

*The World Wide Web has succeeded in large part because its software architecture has been designed to meet the needs of an Internet-scale distributed hypermedia system.*

For Fielding, the terms “(modern) Web” and “(distributed) hypermedia” are inseparable. REST is therefore an architectural style for an (Internet-scaled) distributed hypermedia system (i.e. a Web).

You won’t find clarification for the term “hypermedia” anywhere in the dissertation. This may be one of the reasons why REST is so often misunderstood. It wasn’t until a later[blog article](https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven) that Fielding explained what he meant with hypertext (synonymous with hypermedia):

*When I say hypertext, I mean the simultaneous presentation of information and controls such that the information becomes the affordance through which the user (or automaton) obtains choices and selects actions.*

In today’s Web, we are met with hypermedia on a daily basis by clicking a link on one webpage that forwards us to another, or by opening a document via a link. In this context, hypermedia is about linking information on the Web.

Fielding views hypermedia as a simple generic user interface that allows complex relationships between information to be mapped via links, regardless of the information source and its physical location (e.g. database or webpage, local machine or remote web server).

Motivation

The Web was created in 1989 in a research project at CERN. The researchers developed the first specifications for HTTP, URL and HTML. There were a lot of ideas for enhancing the Web. Numerous enhancements were created. What was missing was a uniform strategy. It quickly became clear that the successful development of the Web would be virtually impossible without standardization.

With REST, they were looking to develop a [standard](https://www.ics.uci.edu/~fielding/pubs/dissertation/evaluation.htm) to describe the way the Web works and serve as a framework for developing Web protocols:

*(…) the motivation for developing REST was to create an architectural model for how the Web should work, such that it could serve as the guiding framework for the Web protocol standards. REST has been applied to describe the desired Web architecture, help identify existing problems, compare alternative solutions, and ensure that protocol extensions would not violate the core constraints that make the Web successful.*

6 Constraints of REST

According to Fielding, REST-compatible architecture should have the following [constraints](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm):

* **1. Client-server architecture:** A server component offers services requested by clients. This is a classic “separation of concerns” principle: Client and server each perform different tasks. In the context of the Web, the client is usually a Web browser. However, the client-server constraint of REST is not limited to Web browsers. Advantage: Client and server can be developed and deployed independently.
* **2. Stateless protocol:** Each client request should contain all necessary information for the server to understand and reply to the request. Servers are not allowed to store specific information about the client and use it for processing a request. In this context, one also speaks of application state. By contrast, the client can save specific information and send it to the server with each request. For this approach, which is exclusive to the client, one speaks of session state. Advantage: Servers can scale better and serve multiple clients.
* **3. Cacheability:** When a server supplies data to a client after a previous request, this data should be marked as cacheable or not cacheable. If the data is cacheable, the client can store it and reuse it as needed without having to request the server again. Advantage: Unnecessary data transmissions are avoided.
* **4. Uniform interface:** When multiple clients are using a server’s services, then communication runs via interfaces that are uniform. This means the server won’t have to offer different types of interfaces depending on client technology. Advantage: the interface remains decoupled from the technical implementation. **According to Fielding, a uniform interface is the most important constraint of REST**. The significance of this constraint requires separate attention and we’ll be exploring it in more detail in a separate blog article.
* **5. Layered system:** This is a hierarchical structure. The components of one layer are only able to communicate with the next respective layer. Communication beyond one layer is not possible. Advantage: A component only needs to know the interface of its adjacent layer.
* **6. Code on demand:** This is the only optional constraint. The client can download code (e.g. JavaScript) and run it locally. Advantage: Client functionality can be expanded without the need for client-side deployment.

Architecture is REST-compatible (i.e. RESTful) when constraints 1 to 5 are met. Fielding refers to this as Uniform-Layered-Client-Cache-Stateless-Server.

Summary

* **REST is a style of architecture for the Web. The central component of REST is a generic user interface based on hypermedia.**