Java SE & Spring

Module 2: Spring

03.Introduction to REST

What is REST?

REST is a set of architectural constraints, not a protocol or a standard. API developers can implement REST in a variety of ways.

When a client request is made via a RESTful API, it transfers a representation of the state of the resource to the requester or endpoint. This information, or representation, is delivered in one of several formats via HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP, or plain text. JSON is the most generally popular programming language to use because, despite its name, it’s language-agnostic, as well as readable by both humans and machines.

What is REST?

REST is acronym for **RE**presentational **S**tate **T**ransfer. It is architectural style for distributed hypermedia systems and was first presented by Roy Fielding in 2000 in his famous dissertation.

Like any other architectural style, REST also does have it's own 6 guiding constraints which must be satisfied if an interface needs to be referred as **RESTful**.

Principles of REST

Client-server – By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.

Stateless – Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client.

Cacheable – Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.

Principles of REST

Uniform interface – By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved. In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behavior of components. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.

Principles of REST

Layered system – The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot “see” beyond the immediate layer with which they are interacting.

Code on demand (optional) – REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be pre-implemented.

Resource

The key abstraction of information in REST is a resource. Any information that can be named can be a resource: a document or image, a temporal service, a collection of other resources, a non-virtual object (e.g. a person), and so on. REST uses a resource identifier to identify the particular resource involved in an interaction between components.

The data format of a representation is known as a media type. The media type identifies a specification that defines how a representation is to be processed. A truly RESTful API looks like hypertext. Every addressable unit of information carries an address, either explicitly (e.g., link and id attributes) or implicitly (e.g., derived from the media type definition and representation structure).

Resource

The key abstraction of information in REST is a resource. Any information that can be named can be a resource: a document or image, a temporal service, a collection of other resources, a non-virtual object (e.g. a person), and so on. REST uses a resource identifier to identify the particular resource involved in an interaction between components.

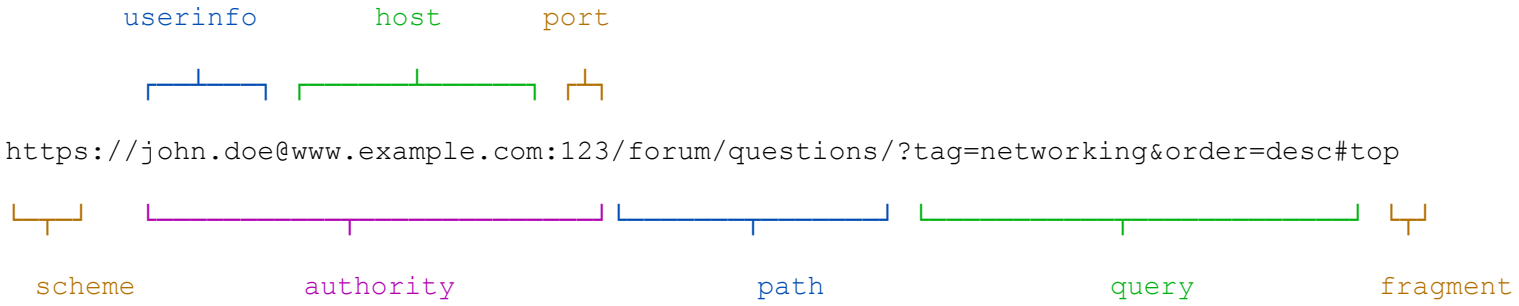
The data format of a representation is known as a media type. The media type identifies a specification that defines how a representation is to be processed. A truly RESTful API looks like hypertext. Every addressable unit of information carries an address, either explicitly (e.g., link and id attributes) or implicitly (e.g., derived from the media type definition and representation structure).

Resource Methods

Method	Description
GET	This method helps in offering **read-only access for the resources**. It consists only header, not body
POST	This method is implemented for **creating a new resource**. It consists both header and body
DELETE	This method is implemented for **removing a resource**. It consists both header and body
PUT	This method is implemented for **updating an existing resource or creating a fresh one**.It consists both header and body

Uniform Resource Identifier

A **URI** (**Uniform Resource Identifier**) is a sequence of characters that identifies a logical or physical resource. Universal Resource Identifiers are specified in the Internet Engineering Task Force (IETF) Request for Comments (RFC) 3986 and are summarized and extended in documentation for the W3C's Web Architecture, Architecture of the World Wide Web, Volume 1.



Further reading: https://datatracker.ietf.org/doc/html/rfc3986

Questions ?

Next:Introduction to Spring Boot