Representational State Transfer

Faculty of Computer Science
Workflow Systems and Technologies

Interoperability SS 2023

Amolkirat Singh Mangat
Matthias Ehrendorfer
Florian Stertz

Representational State Transfer (REST)

- Emerged from Roy Fielding's Dissertation in early 2000
- "***Architectural style****s and the design of network-based software architectures.*"
- Collection of architectural constraints for the behaviour of hypermedia
- Interfaces that fulfil the REST constraints are considered *"RESTful"*
- Hypermedia as the engine of application state (**HATEOAS)**

Architectural Constraints

- *Client Server Model*
  - Enforces separation of concerns
  - Client and server can evolve independently
- *Stateless*
  - Induces visibility, reliability, and scalability
  - May cause network overhead
- Cache
  - Improved efficiency and scalability
  - At the cost of reliability as the cached state starts deviating from the current state over time

Architectural Constraints

- *Uniform interface*
  - Decoupling of services from implementation
  - Loss in efficiency due to the use of standardised non-application specific representations for transfer
  - Key concepts: *identification of resources* and *HATEOAS*
- Layered-System
  - Composition of hierarchical layers
  - Overhead due to latency
- Code-On-Demand
  - Extension of client capabilities on demand

Architectural Data Elements

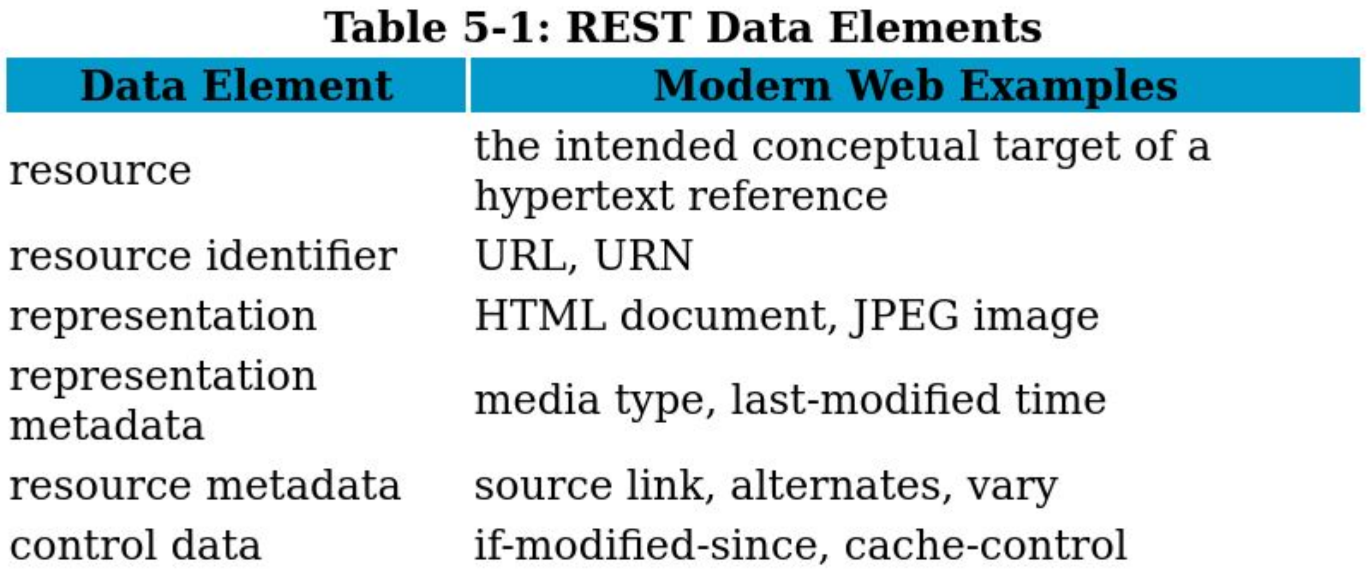


Table 5-1: REST Data Elements

Data Element	Modern Web Examples
resource	the intended conceptual target of a hypertext reference
resource identifier	URL, URN
representation	HTML document, JPEG image
representation metadata	media type, last-modified time
resource metadata	source link, alternates, vary
control data	if-modified-since, cache-control

https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm#tab_5_1

HTTP GET Example

Request

```
GET /interop/attendees HTTP/1.1
Host: univie.interop.at
Accept: application/json, application/xml
```

Response

```
HTTP/1.1 200 OK
Content-Length: <???>
Content-Type: application/xml

<attendees/>
```

HTTP POST Example

Request

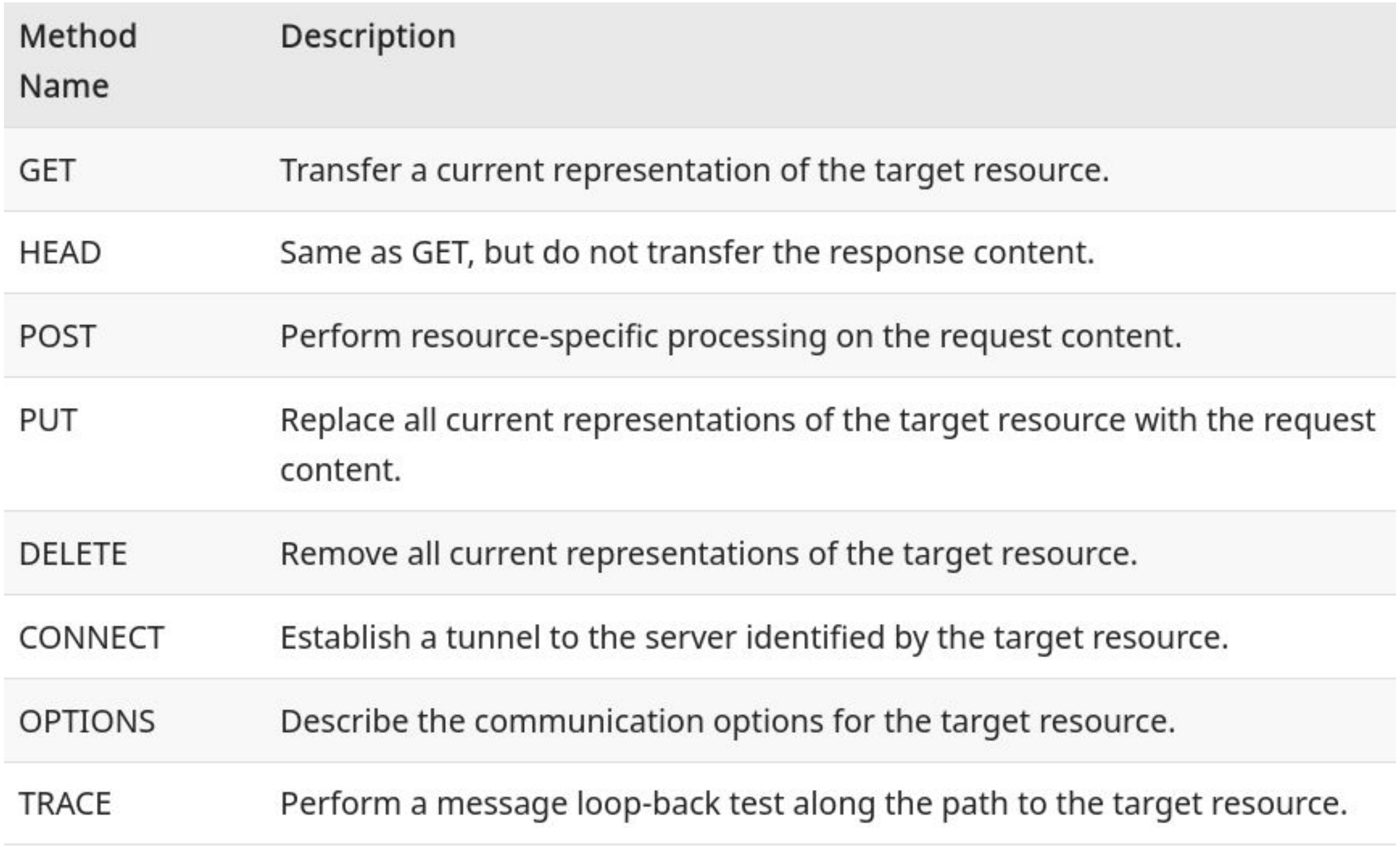
```
POST /interop/attendees HTTP/1.1
Host: univie.interop.at
Content-Type: application/x-www-form-urlencoded

attendee=Alice
```

Response

```
HTTP/1.1 201 OK
Location: /interop/attendee/256
```

Overview of HTTP Methods



Method Name	Description
GET	Transfer a current representation of the target resource.
HEAD	Same as GET, but do not transfer the response content.
POST	Perform resource-specific processing on the request content.
PUT	Replace all current representations of the target resource with the request content.
DELETE	Remove all current representations of the target resource.
CONNECT	Establish a tunnel to the server identified by the target resource.
OPTIONS	Describe the communication options for the target resource.
TRACE	Perform a message loop-back test along the path to the target resource.

https://www.rfc-editor.org/rfc/rfc9110#table-4

HTTP Methods

- HTTP methods describe the *primary semantics* of a request (see RFC 9110).
- Semantics provide a *uniform interface*, i.e. they are not tied to a specific resource. Headers can be applied for additional semantics (e.g. Accept).
- However, a resource determines if those semantics are considered
- RFC 5789 extends set of methods with PATCH
  - Requests a set of changes to be applied on a resource
  - See the approaches in RFC 6902 and RFC 7386
- Methods can be classified into **safe** and/or **idempotent** methods.
  - *Safe* methods don't cause permanent state altering side effects
  - *Idempotent* methods when repeatedly executed result in the same states

HTTP Status Codes

- HTTP status codes (three-digit integer) describes the result and semantics of a response
- Usually a textual description of the status code is also provided
- Classes of status codes:

  1xx Informational: Request was received, continuing process

  2xx Successful: Request was received, understood and accepted

  3xx Redirect: Additional action required to complete the request

  4xx Client Error: Request is malformed or cannot be fulfilled

  5xx Server Error: Failed to fulfill an apparently valid

Hypermedia

Wikipedia (2022) *"... an extension of the term hypertext, is a nonlinear medium of information that includes graphics, audio, video, plain text and hyperlinks."*

Cambridge Dictionary *"a combination of videos, images, sounds, text, etc. that are connected together on a website, which you can click on in order to use them or to go to other related videos, websites, etc."*

HATEOAS

- "Hypermedia is the engine of application state"
- Use of *hypermedia* is a fundamental *constraint*
  - Server must provide *self-descriptive representations* of resources
  - Client can thus interpret the acquired representations and explore its server-provided options regarding obtaining or manipulating the application state
- Prior knowledge beyond the *initial URI* and *standardised media types* used for the server-returned representations shall not be required
- Example:
  - HTML (*Hypertext* Markup Language) uses hyperlinks to provide access to images, web documents etc.
  - Browser (client) understands HTML and knows how to render it and what to do when a hyperlink is triggered

Media Types for Resources

- Multipurpose Internet Mail Extensions (MIME), aka. Media Types
- Indicates format of resources
- Managed by IANA - RFC 6838
- Common standardised Media Types:
  - application/xml
  - application/json
  - text/html
  - application/x-www-form-urlencoded

References

[1] Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures

[2] https://www.rfc-editor.org/rfc/rfc9110

[3] https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven