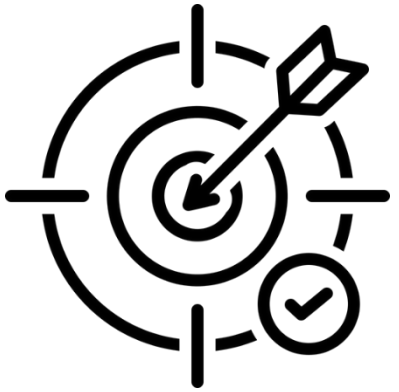
Designing REST Services: HTTP Status Codes

Web Programming and Testing

Mario Simaremare, S.Kom., M.Sc.
Program Studi Sarjana Sistem Informasi
Institut Teknologi Del

Objectives

- The objective of this session is the following:
  - The students are able to develop REST-based services. On this session, we focus on the design phase especially the use of HTTP status codes.

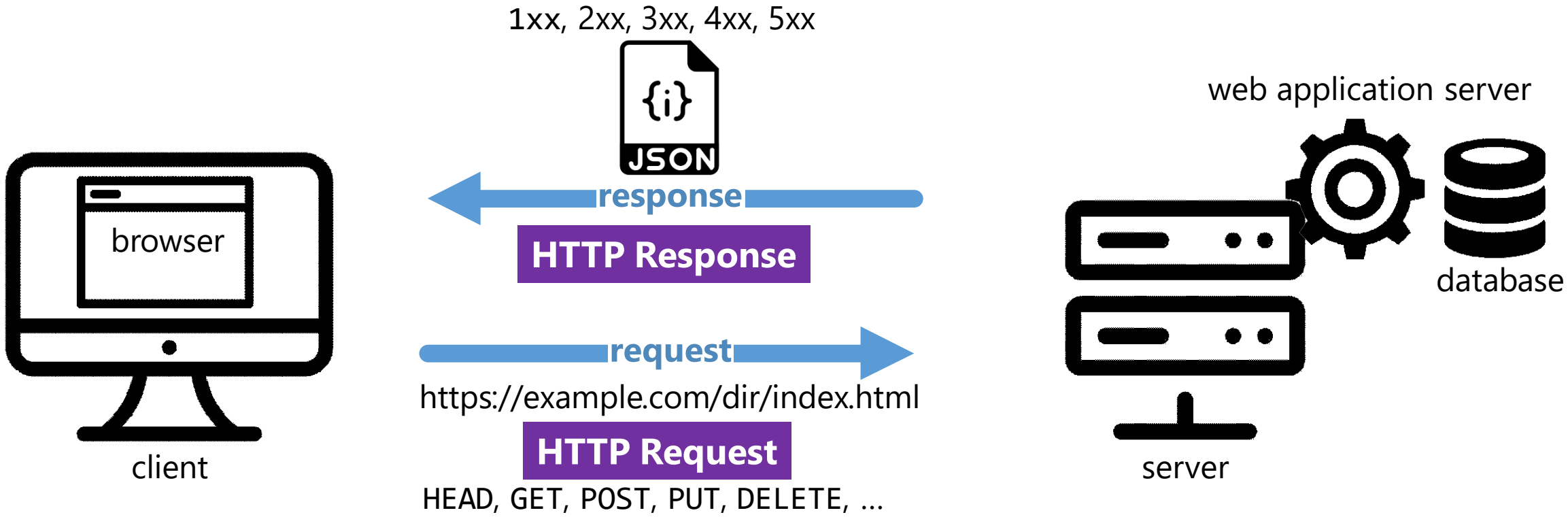


Outlines

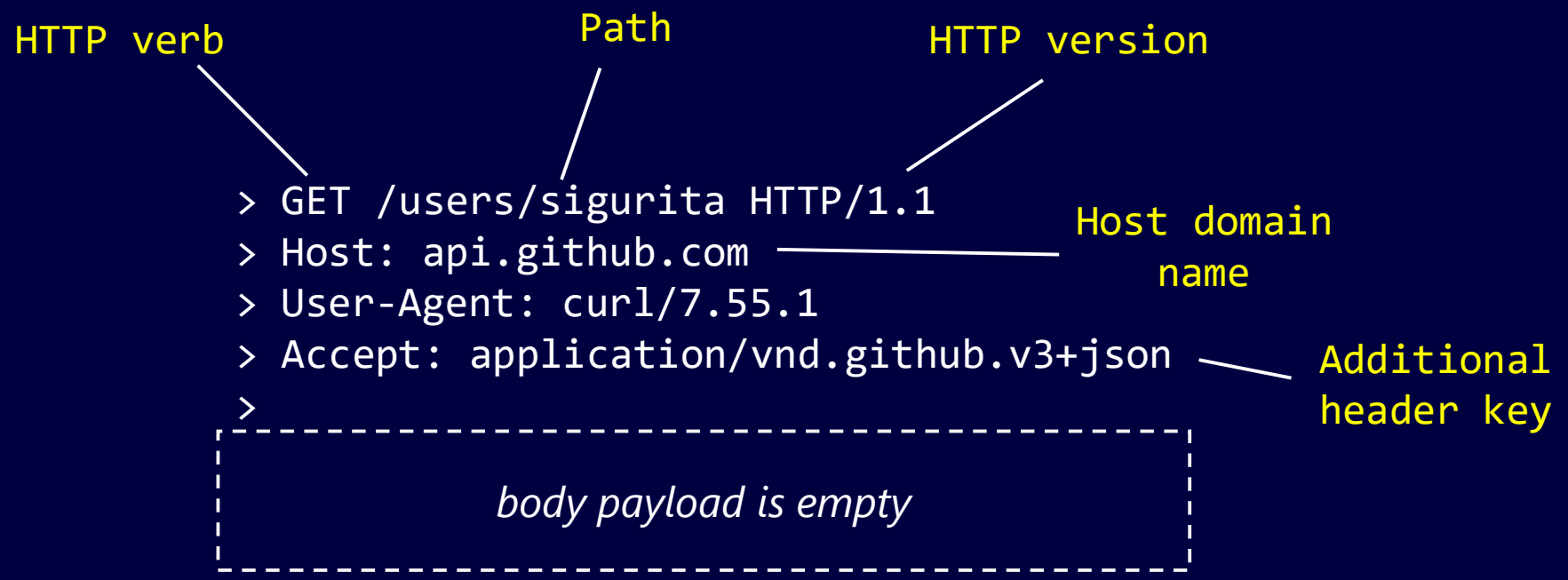
1. Semantic in REST Request-Response Interactivity.
2. HTTP Status Codes.

Semantic in REST
Request-Response Interactivity

Request-Response Cycle

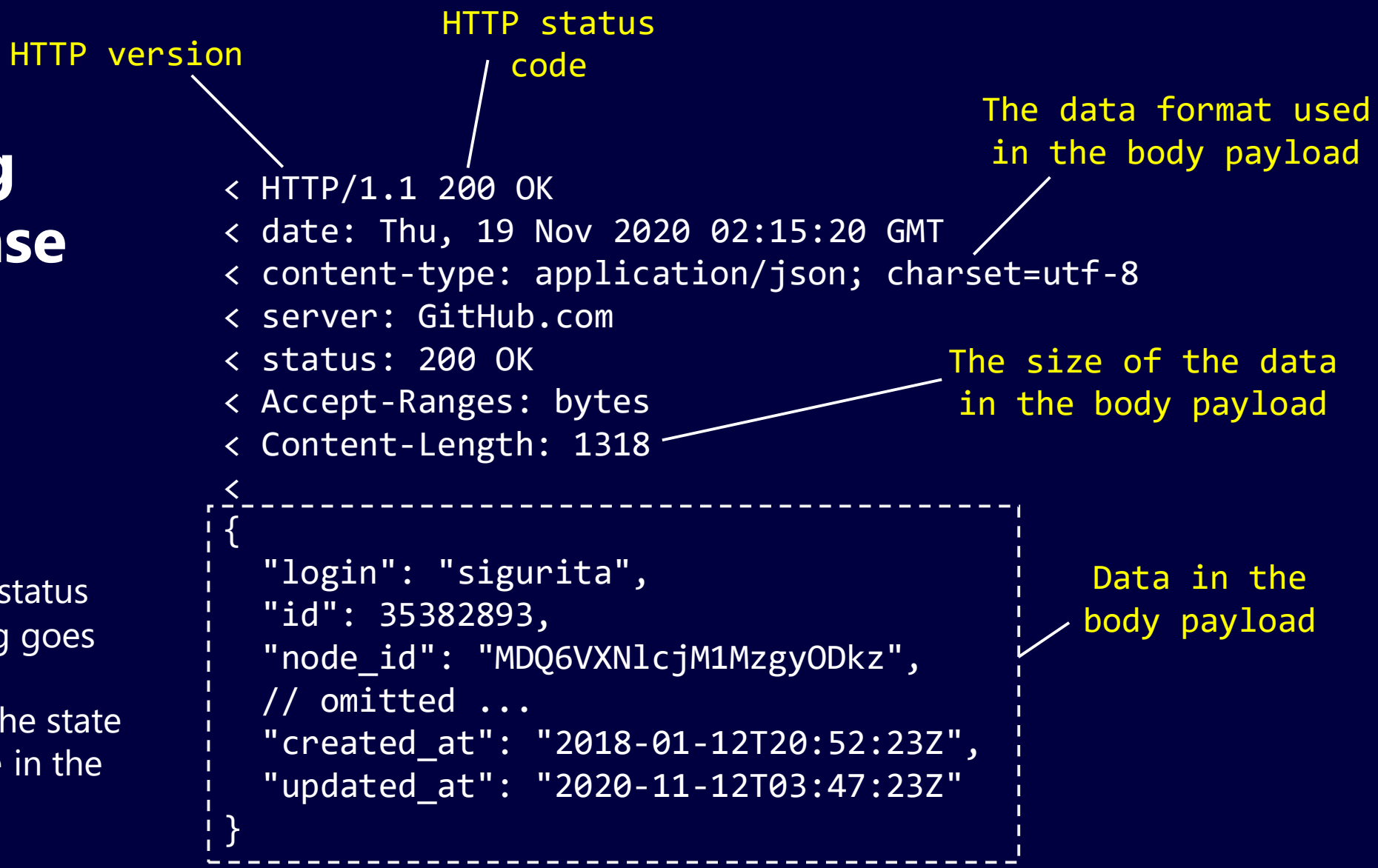


Decomposing HTTP Request



The request uses GET method, meaning it will not cause state change the the requested resource.
In this example, the request does not attach any data in the body payload.

Decomposing HTTP Response



The response retuns 200 status code, meaning everything goes fine (OK).
The response also bring the state of the requested resource in the form of JSON.

HTTP Status Codes

HTTP Status Code

- Every response is summarized in the form of HTTP status code.
  - The status code is added in the response header.

- HTTP status codes:
  - `100` family is used for informational purposes.
  - `200` family is used when things go as expected.
  - `300` family for redirections.
  - `400` family is used to indicate errors caused by the client.
  - `500` family is used to indicate errors caused by the server

Status Code	Meaning
1xx	Informational
2xx	Successful execution
3xx	Redirection
4xx	Unsuccessful request caused by the consumer
5xx	Unsuccessful request caused by the provider

https://www.rfc-editor.org/rfc/rfc2616.txt

HTTP Status Code Rules

- Rule #25: `200` (`OK`) indicates that the request is completely executed without problems.
  - Response to general `GET`, `HEAD`, `DELETE`, and `OPTIONS` methods.

- Rule #26: `201` (`Created`) indicates that a resource is successfully created.
  - Response to `POST` and `PUT` methods.

HTTP Status Code Rules

- Rule #27: `202` (`Accepted`) indicates a successful start of an asynchronous action.
  - It could also mean the request has been inserted to the queue.
  - Used in AJAX call (discussed later).

- Rule #28: `204` (`No Content`) should be used when the payload is intentionally left out.
  - The API decides to not sending any data in the payload.
  - Incomplete or no representational resource available.
  - Used when deleting or removing a resource.

HTTP Status Code Rules

- Rule #29: `301` (`Moved Permanently`) indicates the targeted resource has been moved to another path (URI).
  - A new URI should be returned.
  - The next request should not use the moved URI.

- Rule #30: `302` (`Found`) the URI is working but the requested resource is not available at the moment, it was moved for another path (URI).
  - The URI should work again in the future.
  - It is a general option of `303` or `307`.

HTTP Status Code Rules

- Rule #31: `303` (`See Other`) indicates the URI is working as intended with a URI as the response resource.
  - The consumer may create a new request to the response URI.
  - e.g. when a payment is done then a URI to the receipt resource is returned to the consumer.

HTTP Status Code Rules

- Rule #32: 304 (Not Modified) indicates that the consumer already has the latest version of the resource state.

- Rule #33: 307 (Temporary Redirect) the given URI is not available at the moment as a temporary replacement, another URI is given to the consumer.

HTTP Status Code Rules

- Rule #34: `400` (`Bad Request`) indicates a general failure that is caused by the consumer.
  - e.g. incomplete or incorrect data fails the validation process.

- Rule #35: `401` (`Unauthorized`) indicates that the consumer has no rights to access the resource due its wrong credential.
  - Invalid credential.
  - Expired access token.

HTTP Status Code Rules

- Rule #36: `403` (`Forbidden`) indicates that the consumer has no rights to access the resource due its permission.
  - Incompatible role, permission, and resource.

- Rule #37: `404` (`Not Found`) indicates that the URI is not valid or simply does not exist.

HTTP Status Code Rules

- Rule #38: `405` (`Method Not Allowed`) indicates that the request method is not available for the URI.
  - e.g. invoking a controller URI with GET method.

- Rule #39: `406` (`Not Acceptable`) indicates that the requested media type is not available.
  - e.g. the consumer asks for a representational of a resource in XML, but the provider cannot produce such format.

HTTP Status Code Rules

- Rule #40: `415` (`Unsupported Media Type`) indicates that the payload attach on the request is not supported.
  - e.g. the consumer sends an `XML` formatted payload, but the provider only accept `JSON`.

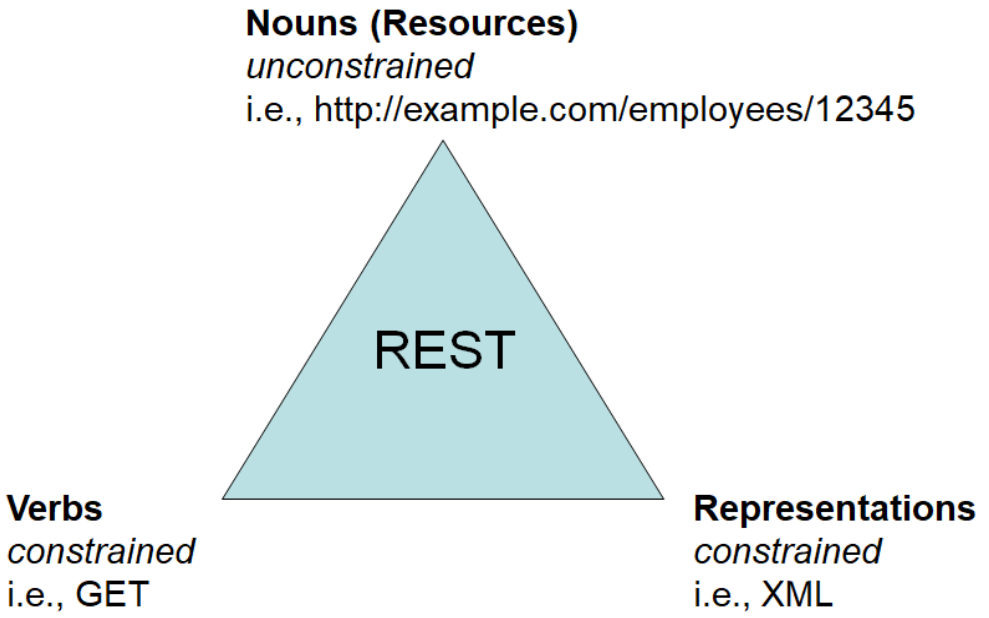
- Some more `4xx` status codes.
  - `409` (`Conflict`).
  - `412` (`Precondition Failed`).
  - etc.

HTTP Status Code Rules

- Rule #41: `500` (`Internal Server Error`) indicates that the API fails due to problems on the provider side.

To-dos

1. Next time we will discuss the representational aspect.
2. Explore the GitHub APIs.



References

Srinivasan, M. (2012). Web Technology: Theory and Practice. Pearson.

Erl T. (2016). Service-Oriented Architecture: Analysis and Design for Services and Microservices. Pearson

Massé, M. (2012). REST API Design Rulebook. O'Reilly

GitHub REST API Documentation. https://docs.github.com/

Thank you