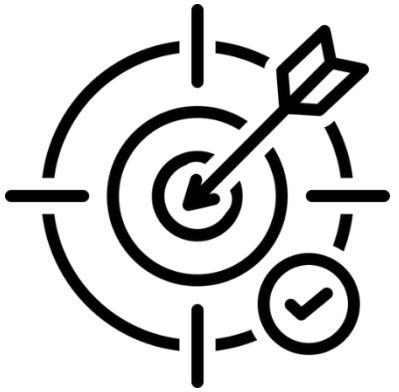
Designing REST Services

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 URI and resource hierarchy.



Outlines

1. Flashback.
2. Design phase.
3. URI Design:
   - General URI Rules.
   - Domain Rules.
   - Resource Modeling.
   - Query String.

Flashback

Service and API Contract

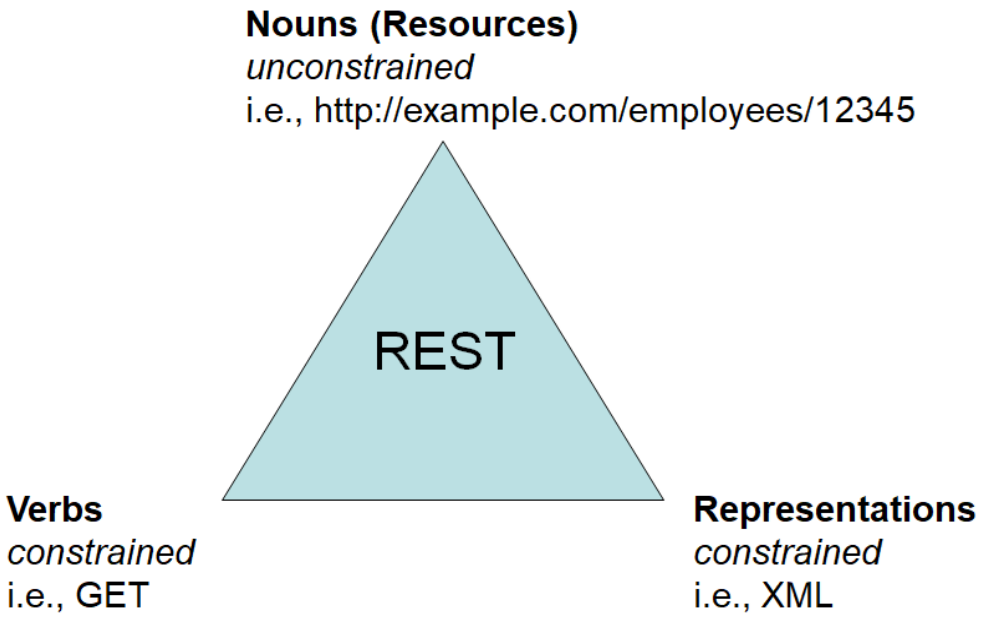
- A service is a software program that makes its functionality available via a published **API** that is part of a service contract.

- The API specification is defined in a contract.
  It contains:
  - How to consume the API,
  - The parameters specifications, and
  - What kind of output is expected.

REST

- REST: REpresentational State Transfer.
  - Is an architectural pattern.
  - It runs on top of the HTTP infrastructure.

- To be discussed:
  - The resource and identifiers.
  - Representation: XML, JSON, etc.
  - Semantics: HTTP verbs (methods) and HTTP response codes.



The Design Phase

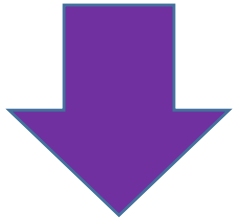
Design Phase is Important

- A service is published for other parties.
  - In some case, the developer does not have any idea how many consumers would actually use it.

- **Changing** a published service could be a true **nightmare** for the consumers sometimes.
  - A carefully crafted service would last.
  - However, a changing behavior is a normal thing.

Fatal Changes

```
transferFund(String _target, BigInteger _amount){
   ...
}
```



```
transferFund(BigInteger _amount, String _target){
   ...
}
```

URI vs. URL

- Unified Resource Identifier.
  - **Resource**: any concept or interact-able object.
  - **Resource Identifier**: a unique naming given to a particular resource that make it addressable.
  - **URI**: a format to uniquely identify resources.

- What is URL? How it is different to URI?
  - URI is about the resource naming.
  - URL is about how to access the resource.

URI Format

- Services are designed for machines to use and humans to understand.

```
scheme://domainname/path[?query][#fragment]
```

- URI format:
  - The `scheme` could be `HTTP/S`, `FTP`, etc.
  - The `query` and `fragment` are optional.

URI: For Machines or Humans?

- Look at the following examples:

```
api.lib.si-playground.com/books/1449310508
```

```
api.lib.si-playground.com/7e1a-43da-4925-8238-d579
```

- Which one of the above URIs is …
  - Suitable for machines?
  - Suitable for humans?
  - What about for both? Machines and humans?

URI Design: General URI Rules

General URI Rules

- Rule #1: Forward slash "/" indicates resource hierarchical relationships.

```
api.lib.si-playground.com/books/1449310508/chapters
```

```
api.github.com/repos/sigurita/hello-world/commits
```

- Rule #2: No trailing forward slash "/".

```
api.lib.si-playground.com/books/1449310508/borrow/
```

```
api.github.com/repos/sigurita/hello-world/commits/
```

General URI Rules

- Rule #3: Hyphens "-" for a better readability.

```
api.lib.si-playground.com/books?q=restful-api
```

- Rule #4: Underscore "_" should be avoided.
  - Visually, underscore is "unseen" in hyperlink output.

A_hyperlink_with_underscores

General URI Rules

- Rule #5: Use lowercase in the URI **path** for consistency.

```
api.lib.si-playground.com/books/1449310508   1
API.LIB.SI-PLAYGROUND.COM/books/1449310508   2
api.lib.si-playground.com/Books/1449310508   3
```

  - #1 is fine.
  - #2 is identical to #1.
  - #3 is not identical to neither #1 nor #2.

General URI Rules

- Rule 6: Extension should not be included.

```
api.lib.si-playground.com/books/1449310508.json
```

URI Design: Domain Rules

Domain Rules

- Rule #7: Consistent subdomain names.
  - Changing subdomain will surely affect the existing endpoints.

```
api.lib.si-playground.com
```

- Rule #8: Consistent subdomains names for developer portal (documentations is essential in REST-based services).

```
docs.github.com
```

URI Design: Resource Modeling

Resource Hierarchy

- There are four types of resource:
  - Document: a unique record or an instance.
  - Collection: a grouped of documents (like directory) managed by the service provider.
  - Store: similar to collection but managed by the service consumer.
  - Controller:

Resource Hierarchy: Document

- A document represents a unique record or an instance.
  - It may contains a sub-concept or resources.
  - e.g. a specific book is a document, yet it has multiple chapters in it. Each chapter is a distinct sub-concept.

```
api.lib.si-playground.com/books/1449310508
```

```
api.lib.si-playground.com/books/1449310508/chapters/3
```

Resource Hierarchy: Collection

- A collection is a resource 'directory' managed by the provider.
  - It may contains a sub-collection.

```
api.lib.si-playground.com/books
```

  - An example.
    - A library provides a collection of book information wrapped in the `/books` path.
    - A book author may offer his/her new book information to be stored in the collection.
    - Since the collection is managed by the provider, the consumer has no control over the acceptance of the offer.

Resource Hierarchy: Store

- A store is a resource 'directory' managed by the consumer.
  - Similar to collection, a store may contains a sub-store.

```
api.lib.si-playground.com/books/1449310508/chapters
```

  - An example.
    - A book author may add or remove any chapter of his/her book registered in the library.
    - The author has a complete control over its resource.

Resource Hierarchy: Controller

- A controller is a procedural concept or action executed on the resource-level.
  - It specifies what the consumer could do over the resource.

```
POST /books/1449310508/request
```

```
PATCH /books/1449310508/borrow
```

```
PATCH /books/1449310508/return
```

Resource Naming Rules

- Rule #9: Singular noun for document.
- Rule #10: Plural noun for collection.
- Rule #11: Plural noun for store.
- Rule #12: Verb or verb phrase for controller.
- Rule #13: Identity may be used in the store.

```
api.lib.si-playground.com/books/{ISBN-10}/chapters
```

Resource Naming Rules

- Rule #14: CRUD controller should be avoided.
  - This example should not be followed.

```
api.lib.si-playground.com/books/1449310508/update
```

  - Instead, use the corresponding HTTP methods.

```
PUT /books/1449310508
```

URI Design: Query String

Query String Rules

- Query is used for specifying extra parameters while consuming a service.
  - It is occasionally used for document filtering.
  - It is located right after the path section.
  - Behind the controller.

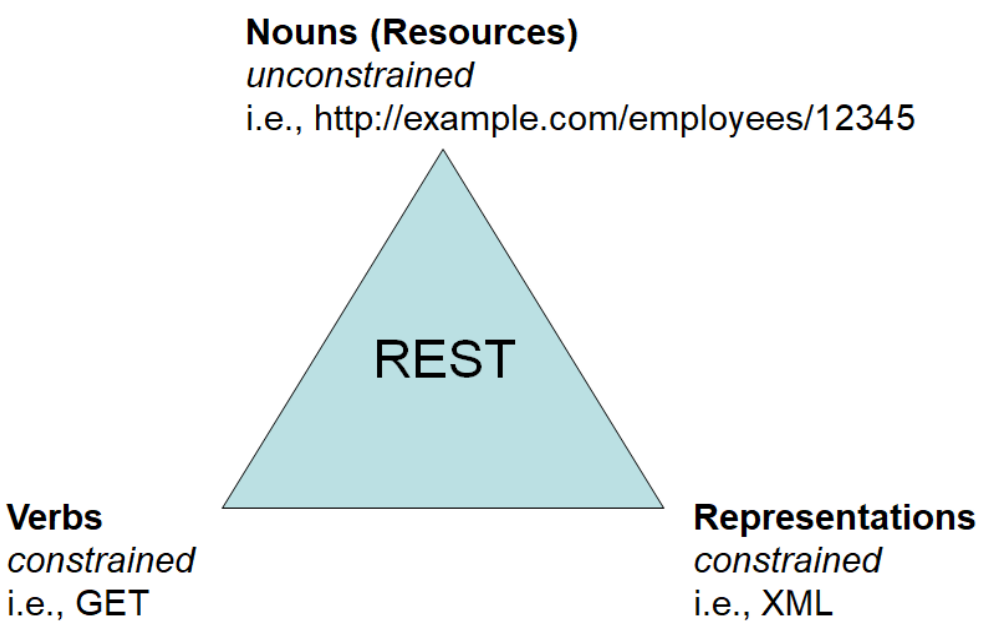
```
scheme://domainname/path[?query][#fragment]
```

- Rule #15: Use the query to set the pagination, filtering criteria, or some other things.

```
GET api.lib.si-playground.com/books?q=restful-api&page-size=10
```

To-dos

1. Next time we will discuss more on the semantic aspects, both the HTTP request verbs and HTTP response codes.
2. Afterwards, we will discuss about the representational aspect.
3. 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