Universitat de Lleida
Departament d'Informàtica
i Enginyeria Industrial

Programming and Communications III: REST & RESTful

Jordi Ricard Onrubia Palacios

Departament d'Informàtica i Enginyeria Industrial Universitat de Lleida

Universitat de Lleida
Departament d'Informàtica
i Enginyeria Industrial

Programming and Communications III: REST Overview

REST - Overview

- REST stands for REpresentational State Transfer.
- It means when a RESTful API is called, the server will transfer to the client a representation of the state of the requested resource.
- For example, when a developer calls Instagram API to fetch a specific user (the resource), the API will return the state of that user, including their name, the number of posts that user posted on Instagram so far, how many followers they have, and more.
- Usually, the representation of the state is done in JSON format, although, we can use other representations such as XML.

REST - Main Points

- Statelessness: Each request from a client to a server must contain all the information needed to process the request; the server does not store client context.
- Client-Server Separation: The client (front-end) and server (back-end) are separate entities, allowing independent evolution.
- Uniform Interface: A consistent way to interact with resources, typically through standard HTTP methods (GET, POST, PUT, DELETE).
- Cacheability: Responses must define whether they can be cached or not to improve performance.
- Layered System: Architecture can be composed of hierarchical layers to improve scalability.
- Code on Demand (optional): Servers can extend client functionality by sending executable code.

REST - Main Points

- Requests must include individual resource identification if needed. Representation is then transferred (JSON, HTML, etc.)
- Client only has representation to work with (must include then all needed data and metadata)
- Messages self-contained (MIME, cache information, encoding, etc.)
- HATEOAS (Hypermedia As The Engine Of Application State). Actions are identified only through hyperlinks (except entry points)

Universitat de Lleida
Departament d'Informàtica
i Enginyeria Industrial

Programming and Communications III: RESTful

RESTful - Overview

RESTful refers to a web service or API that adheres to the principles of REST (Representational State Transfer). It is an implementation of REST principles designed to manage resources over the internet using standard protocols like HTTP.

RESTful - Main Points

1. Resource-Oriented Design: The focus is on resources, which are represented as URLs (Uniform Resource Locators)
2. Stateless Communication
3. HTTP Verbs for Operations
4. Uniform Interface: A consistent and standardized way of accessing and manipulating resources.
5. Representations of Resources: Resources can be represented in multiple formats, such as JSON, XML, or plain text.

RESTful - Main Points

6. HATEOAS (Optional in Practice): Hypermedia as the Engine of Application State: Responses contain links to other related actions or resources, guiding the client dynamically.
7. Cacheability: Responses must explicitly define whether they can be cached, improving performance.

RESTful - HATEOAS

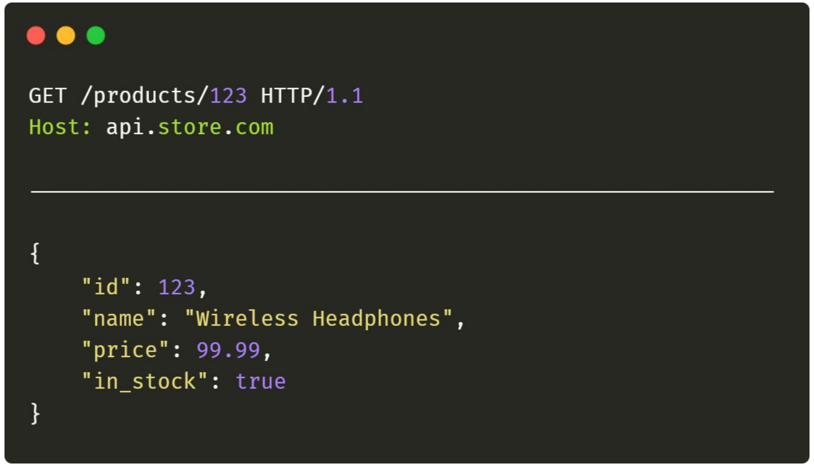
HATEOAS (Hypermedia As The Engine Of Application State) is a key concept in the REST (Representational State Transfer) architecture. It ensures that a RESTful API dynamically guides the client on how to interact with the system through links embedded in responses. This makes the API self-descriptive and allows clients to discover available actions dynamically without relying on hardcoded paths or external documentation.

RESTful - HATEOAS

- Hypermedia-Driven:
  - Hypermedia (e.g., links, forms) is used in API responses to navigate and perform actions.
  - Clients don't need to know the exact structure of the API beforehand.
- Dynamic Interaction:
  - The server provides links in the response to related resources or actions the client can take.
- Self-Descriptive Messages:
  - Each response contains enough information for the client to understand the state of the resource and how to transition to other states.

RESTful - HATEOAS

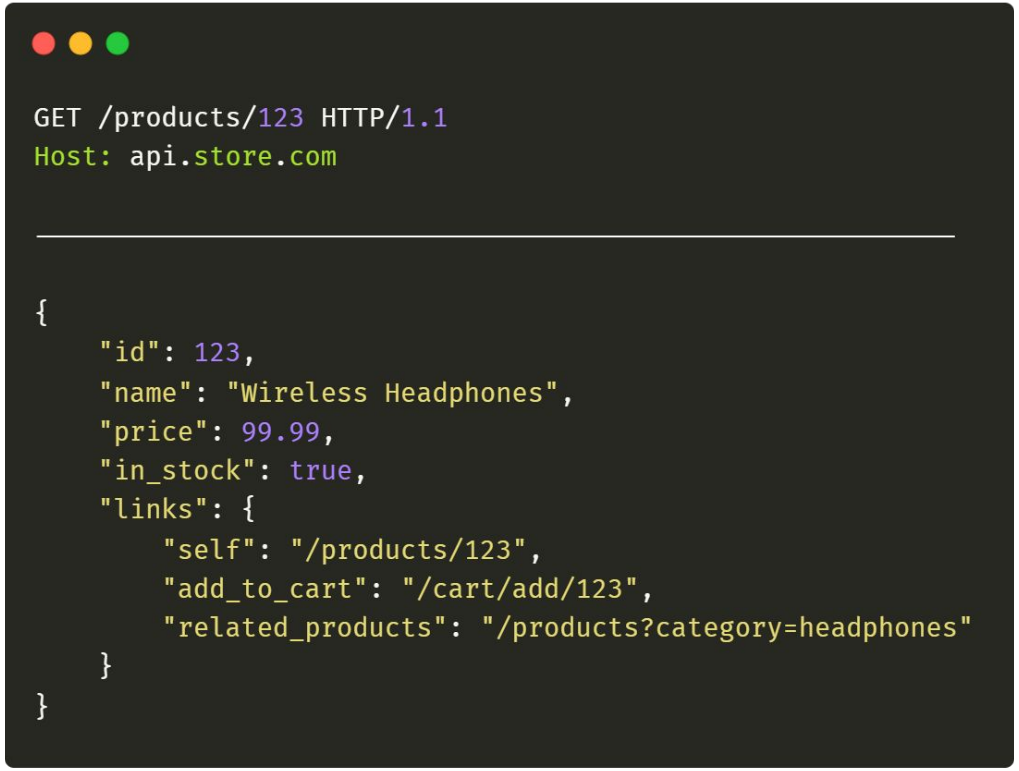
Without HATEOAS



```
GET /products/123 HTTP/1.1
Host: api.store.com
```

```
{
    "id": 123,
    "name": "Wireless Headphones",
    "price": 99.99,
    "in_stock": true
}
```

With HATEOAS



```
GET /products/123 HTTP/1.1
Host: api.store.com
```

```
{
    "id": 123,
    "name": "Wireless Headphones",
    "price": 99.99,
    "in_stock": true,
    "links": {
        "self": "/products/123",
        "add_to_cart": "/cart/add/123",
        "related_products": "/products?category=headphones"
    }
}
```

RESTful - Endpoint/URIs

An endpoint in the context of web services or APIs is a specific URL or URI (Uniform Resource Identifier) that represents a resource or functionality that a client can interact with on a server. Endpoints are the access points through which an API exposes its services.

RESTful - Endpoint/URIs

1. Base URL:
   1.1. The root address of the API.
   1.2. Example: https://api.example.com.
2. Path:
   2.1. Defines the specific resource or action.
   2.2. Example: /users/123 (accessing the user with ID 123).
3. HTTP Method

RESTful - Endpoint/URIs

4. Query Parameters (optional):
   4.1. Provides additional filters or options.
   4.2. Example: ?sort=asc&limit=10.
5. Headers (optional): Additional metadata sent with the request, like authentication tokens or content types.
6. Body (Specific cases such as POST, PUT): Includes all the data required by the server to operate.

RESTful - Endpoint/URIs

- Unique: Each endpoint represents a unique resource or action.
- Consistent: Well-designed APIs use consistent endpoint structures for simplicity.
- RESTful APIs:
  - Endpoints are often resource-based (nouns like /users, /products).
  - Actions are determined by HTTP methods, not by the endpoint path (e.g., /users + GET/POST/PUT/DELETE).

RESTful - Endpoint/URIs

Path	GET	POST	PUT	DELETE
/clients/	List Clients	-	-	-
/clients/{id}	Client {id} data	Create client {id}	Modify client {id}	Delete client {id}
/clients/{id}/{addrid}	Client {id} address {addrid} data	Add address {addrid} to client {id}	Modify address {addrid} to client {id}	Delete address {addrid} from client {id}
/clients/{id}/orders/	Client {id} order list		-	-
/clients/{id}/orders/{orderid}	Client {id} order {orderid} data	Add order {orderid} to client {id}	Modify order {orderid} to client {id}	Delete order {orderid} from client {id}

Universitat de Lleida
Departament d'Informàtica
i Enginyeria Industrial

Programming and Communications III: Conclusions

❑ REST
❑ RESTful
➢ Conclusions

Conclusions

- REST: A high-level design philosophy or set of principles that provides guidelines for designing systems without specifying implementation details.
- RESTFul: Systems or APIs that implement REST principles, refers to the practical implementation of REST guidelines, RESTful APIs are built using tools and frameworks.