REST APIs

[Shimaa Adel Mohammed Elzarief]

REST Constraints

1. **Uniform interface**

   Requests from different clients look the same, whether the client is a chrome browser, a linux server, a python script, an android app or anything else.

2. **Client — server separation**

   The client and the server act independently, each on its own, and the interaction between them is only in the form of requests, initiated by the client only, and responses, which the server send to the client only as a reaction to a request. The server just sits there waiting for requests from the client to come. The server doesn’t start sending away information about the state of some resources on its own.

3. **Stateless**

   The server does not remember anything about the user who uses the API. It doesn’t remember if the user of the API already sent a GET request for the same resource in the past, it doesn’t remember which resources the user of the API requested before, and so on.

4. **Layered system**

   Between the client who requests a representation of a resource’s state, and the server who sends the response back, there might be a number of servers in the middle. These servers might provide a security layer, a caching layer, a load-balancing layer, or other functionality. Those layers should not affect the request or the response. The client is agnostic as to how many layers, if any, there are between the client and the actual server responding to the request.

5. **Cacheable**

This means that the data the server sends contain information about whether or not the data is cacheable. If the data is cacheable, it might contain some sort of a version number. The version number is what makes caching possible: since the client knows which version of the data it already has (from a previous response), the client can avoid requesting the same data again and again. The client should also know if the current version of the data is expired, in which case the client will know it should send another request to the server to get the most updated data about the state of a resource.

6. **Code-on-demand**

This constraint is optional — an API can be RESTful even without providing code on demand.The client can request code from the server, and then the response from the server will contain some code, usually in the form of a script, when the response is in HTML format. The client then can execute that code.

[Burham Soliman]

- *What is a RESTful API ?*
    - It could be defined as " Architectural style that defines a set of constraints to be used for creating a web services " [1]
    - It's a simple and flexible way of accessing web services without having any processing.

- *How does the RESTful API work ?*

    - The RestAPI mechanism is so simple. It starts when a client sends a request to the server in the form of a web URL using one of the HTTP requests (POST - GET - PUT - PATCH - DELETE). [2]
    - In return, the server responds to this request with simple data which can be (XML - HTML - Image - JSON, etc...).
    - All formats are expected as per the request, but FYK that now a days , JSON is the most used format in such responses

- ## *Why should you use RESTful API ?*
    - With no doubt as a developer you're always looking for the best way to achieve your goals with the greatest performances and less code as much as it can be.
    - So that's what you can achieve with RESTful API, as long as you're going to gain the expected results from your work with less bandwidth usage, and faster response to your request, with some of constraints that controls the whole process.
        - ### Advantages of RESTful API ?
            - Easy to learn and understand based on its simplicity.
            - Using less bandwidth compared with other protocols.
            - Scalable and flexible according to the request itself.
            - Independency, as long as there are no contracts between servers and clients.

- ## *What are the RESTful API 's disadvantages?* [3]
    - As per what i mentioned before, the RestAPI has no contracts between the server and clients, it has to be communicated by other means such as E-Mails.

    - As a HTTP service, the were no Asynchronous calls
    - Active sessions can’t be controlled or determined.

—-

[References]

(1) Redhat Topics
(2) Geeks4Geeks
(3) JournalDev