HTTP Methods. REST & JSON.

PRAGMATIC IT Learning & Outsourcing Center

www.pragmatic.bg

Lector: Stefan Vadev
Skype: stefanvadev77
E-mail: stefan.vadev@gmail.com
Facebook:
https://www.facebook.com/stefan.vadev.7

Summary

- HTTP methods
- What is REST
- Why REST is the best
- How to use REST
- JSON
- Workshop

Remember what is HTTP?

Remember what is HTTP?

What is HTTP?
The **Hypertext Transfer Protocol (HTTP)** is designed to enable communications between clients and servers.

HTTP works as a request-response protocol between a client and server.

A web browser may be the client, and an application on a computer that hosts a web site may be the server.

Example: A client (browser) submits an **HTTP** request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

Main HTTP methods

Two HTTP Request Methods: GET and POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

GET - Requests data from a specified resource
POST - Submits data to be processed to a specified resource

GET Method

GET requests can be cached

GET requests remain in the browser history

GET requests can be bookmarked

GET requests should never be used when dealing with sensitive data

GET requests have length restrictions

GET requests should be used only to retrieve data

POST Method

POST requests are never cached

POST requests do not remain in the browser history

POST requests cannot be bookmarked

POST requests have no restrictions on data length

Let's compare GET & POST

https://www.w3schools.com/tags/ref_httpmethods.asp

Other HTTP methods

HEAD
- Same as GET but returns only HTTP headers and no document body

PUT
- Uploads a representation of the specified URI

DELETE
- Deletes the specified resource

OPTIONS
- Returns the HTTP methods that the server supports

CONNECT
- Converts the request connection to a transparent TCP/IP tunnel

REST

{ REST }

What is REST

REPRESENTATIONAL STATE TRANSFER

REST, is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other. REST-compliant systems, often called RESTful systems, are characterized by how they are stateless and separate the concerns of client and server. We will go into what these terms mean and why they are beneficial characteristics for services on the Web.

What is REST

SEPARATION OF CLIENT AND SERVER

In the REST architectural style, the implementation of the client and the implementation of the server can be done independently without each knowing about the other. This means that the code on the client side can be changed at any time without affecting the operation of the server, and the code on the server side can be changed without affecting the operation of the client.

Statelessness

Systems that follow the REST paradigm are stateless, meaning that the server does not need to know anything about what state the client is in and vice versa. In this way, both the server and the client can understand any message received, even without seeing previous messages. This constraint of statelessness is enforced through the use of resources, rather than commands. Resources are the nouns of the Web - they describe any object, document, or thing that you may need to store or send to other services.

Statelessness

Systems that follow the REST paradigm are stateless, meaning that the server does not need to know anything about what state the client is in and vice versa. In this way, both the server and the client can understand any message received, even without seeing previous messages. This constraint of statelessness is enforced through the use of resources, rather than commands. Resources are the nouns of the Web - they describe any object, document, or thing that you may need to store or send to other services.

Client <-> Server communication

In the REST architecture, clients send requests to retrieve or modify resources, and servers send responses to these requests. Let’s take a look at the standard ways to make requests and send responses.

MAKING REQUESTS

REST requires that a client make a request to the server in order to retrieve or modify data on the server. A request generally consists of:

- an HTTP verb, which defines what kind of operation to perform
- a header, which allows the client to pass along information about the request
- a path to a resource
- an optional message body containing data

Headers and Accept parameters

HEADERS AND ACCEPT PARAMETERS
In the header of the request, the client sends the type of content that it is able to receive from the server. This is called the Accept field, and it ensures that the server does not send data that cannot be understood or processed by the client. The options for types of content are MIME Types (or Multipurpose Internet Mail Extensions, which you can read more about in the MDN Web Docs.

MIME Types, used to specify the content types in the Accept field, consist of a type and a subtype. They are separated by a slash (/).

Headers and Accept parameters

For example, a text file containing HTML would be specified with the type text/html. If this text file contained CSS instead, it would be specified as text/css. A generic text file would be denoted as text/plain. This default value, text/plain, is not a catch-all, however. If a client is expecting text/css and receives text/plain, it will not be able to recognize the content.

Other types and commonly used subtypes:
image — image/png, image/jpeg, image/gif
audio — audio/wav, image/mpeg
video — video/mp4, video/ogg
application — application/json, application/pdf, application/xml, application/octet-stream

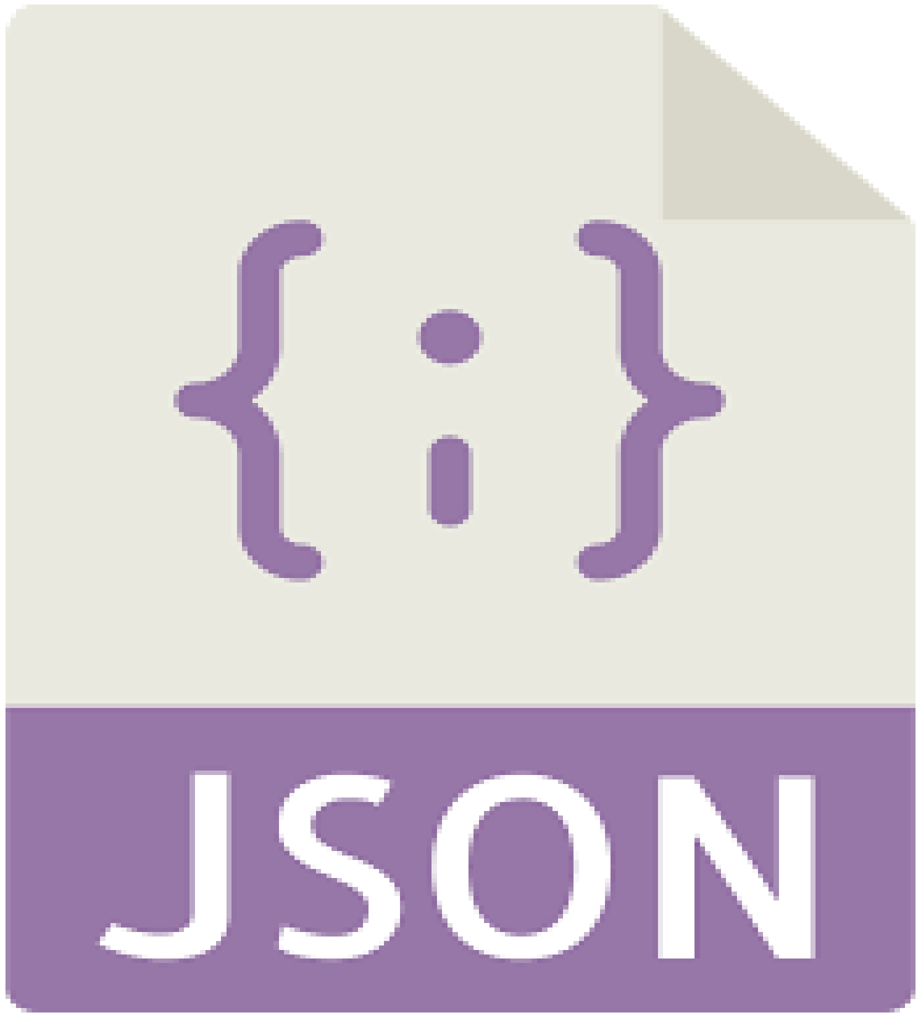
More info about REST here:

https://www.codecademy.com/articles/what-is-rest

http://www.restapitutorial.com/

https://spring.io/understanding/REST

JSON



What is JSON

JSON stands for JavaScript Object Notation

JSON is a lightweight data-interchange format

JSON is "self-describing" and easy to understand

JSON is language independent

More about JSON

JSON is used when Exchanging Data
When exchanging data between a browser and a server, the data can only be text.

JSON is text, and we can convert any Java object into **JSON**, and send **JSON** to the server.

We can also convert any **JSON** received from the server into Java objects.

This way we can work with the data as Java objects, with no complicated parsing and translations.

JSON examples

```
{ "name":"John", "age":30, "car":null }
```

JSON schema

```
{
   "title": "Person",
   "type": "object",
   "properties": {
      "firstName": {
         "type": "string"
      },
      "lastName": {
         "type": "string"
      },
      "age": {
         "description": "Age in years",
         "type": "integer",
         "minimum": 0
      }
   },
   "required": ["firstName", "lastName"]
}
```

More info about JSON here

https://www.w3schools.com/js/js_json_intro.asp

https://www.tutorialspoint.com/json/index.htm

Workshop

https://spring.io/guides/gs/spring-boot/

https://spring.io/guides/gs/rest-service/

https://spring.io/guides/gs/consuming-rest/

www.pragmatic.bg

PRAGMATIC IT Learning & Outsourcing Center

Summary

- HTTP methods
- What is REST
- Why REST is the best
- How to use REST
- JSON
- Workshop