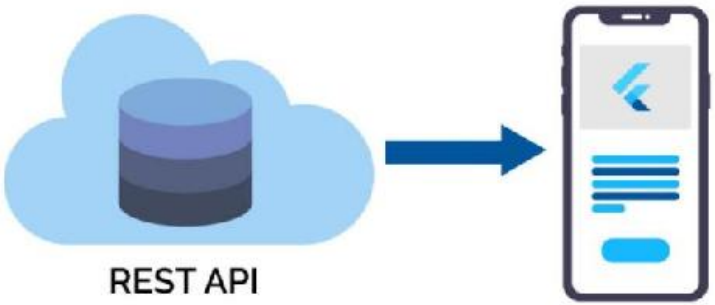
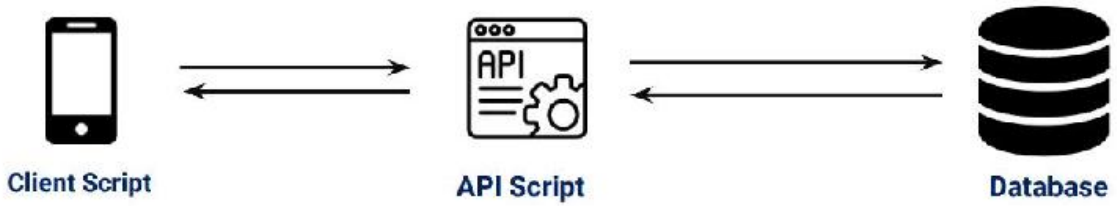
REST API Concept

Flutter With Web Rest API



Introduction to Rest API

Representational State Transfer



Postman

Rest API practices

- Best practices of json
- Request response model
- Rest API working flow
- Postman & Documentation

Practices of JSON

JavaScript Object Notation (JSON)

- JSON is a lightweight data-interchange format that is completely language independent.
- It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data
- The official Internet media type for JSON is application/json.
- It was designed for human-readable data interchange.
- The filename extension is .json.

Practices of JSON

Uses of JSON

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.

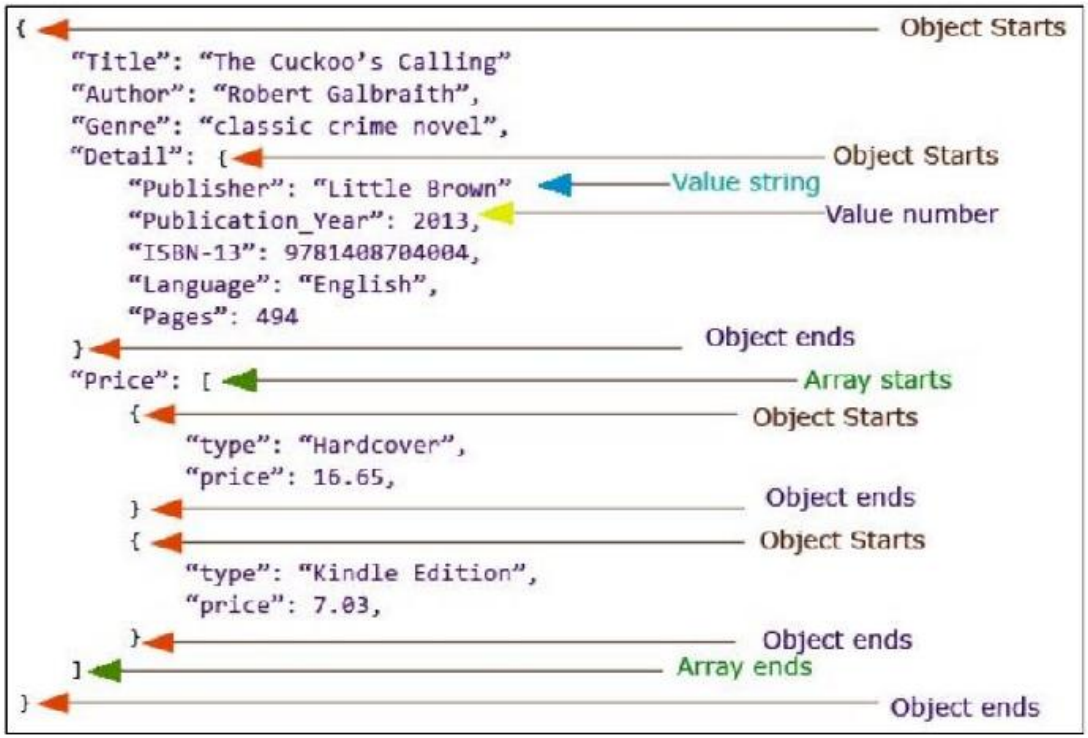
Practices of JSON

Characteristics of JSON

- JSON is easy to read and write.
- It is a lightweight text-based interchange format.
- JSON is language independent.

Practices of JSON

Understanding JSON Structure



Practices of JSON

JSON - Data Types

Type	Description
Number	Double- precision floating-point format in JavaScript
String	Double-quoted Unicode with backslash escaping
Boolean	True or False
Array	An ordered sequence of values
Value	It can be a string, a number, true or false, null etc
Object	An unordered collection of key:value pairs
Whitespace	Can be used between any pair of tokens
null	Empty

Practices of JSON

- Always enclose the **Key : Value** pair within double quotes

```
{'id': '1','name':File} is not right ✗

{"id": 1,"name":"File"} is okay  ✓

{"id": "1","name":"File"} is the best  ✓
```

Practices of JSON

- Never use Hyphens in your Key fields

```
{"first-name":"Rachel","last-name":"Green"}  is not right. ✗

{"first_name":"Rachel","last_name":"Green"} is okay ✓

{"firstname":"Rachel","lastname":"Green"} is okay ✓

{"firstName":"Rachel","lastName":"Green"} is the best. ✓
```

Practices of JSON

Bad Special Characters & Solution

Characters	Replace with
Backspace	\b
Form feed	\f
Newline	\n
Carriage return	\r
Tab	\t
Double quote	\"
Backslash	\\

Practices of JSON

- Bad Special Characters And Solution

```
"<h2>About US <br>\r\n</h2>\r\n<p>It has been exactly 3 years since I wrote my first blog series
\r\nentitled “Flavorful Tuscany”, but starting it was definitely not easy. \r\nBack then, I didn’t know
much about blogging, let alone think that one \r\nday it could become <strong>my full-time job</strong>.
Even though I had\r\n many recipes and food-related stories to tell, it never crossed my mind\r\n that I
could be sharing them with the whole world.</p>\r\n<p>I am now a <strong>full-time blogger</strong> and
the curator of the <a data-cke-saved-href=\"https://ckeditor.com/ckeditor-4/#\"
href=\"https://ckeditor.com/ckeditor-4/#\">Simply delicious newsletter</a>, sharing stories about
traveling and cooking, as well as tips on how to run a successful blog.</p>\r\n<p>If you are tempted by
the idea of creating your own blog, please think about the following:</p>\r\n<ul><li>Your story (what do
you want to tell your audience)</li><li>Your audience (who do you write for)</li><li>Your blog name and
design<br>\r\n\t</li></ul>\r\n<p>After you get your head around these 3 essentials, all you have to do
is grab your keyboard and the rest will follow.</p>"
```

Practices of JSON

- Always create a Root element.

JSON with root element

```
{
"menu": [
  {
    "id": "1",
    "name":"File",
    "value": "F",
    "popup": {
      "menuitem": [
        {"name":"New", "value": "1N", "onclick": "newDoc()"},
        {"name":"Open", "value": "1O", "onclick": "openDoc()"},
        {"name":"Close", "value": "1C", "onclick": "closeDoc()"}
        ]
      }
  },
  {
    "id": "2",
    "name":"Edit",
    "value": "E",
    "popup": {
      "menuitem": [
        {"name":"Undo", "value": "2U", "onclick": "undo()"},
        {"name":"Copy", "value": "2C", "onclick": "copy()"},
        {"name":"Cut", "value": "2T", "onclick": "cut()"}
        ]
      }
  }
  ]
}
```

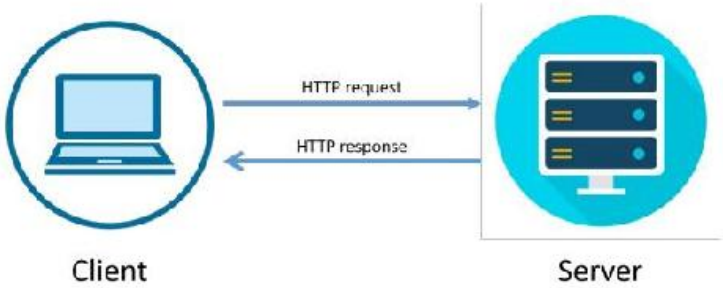
JSON without root element

```
[
  {
    "id": "1",
    "name":"File",
    "value": "F",
    "popup": {
      "menuitem": [
        {"name":"New", "value": "1N", "onclick": "newDoc()"},
        {"name":"Open", "value": "1O", "onclick": "openDoc()"},
        {"name":"Close", "value": "1C", "onclick": "closeDoc()"}
        ]
      }
  },
  {
    "id": "2",
    "name":"Edit",
    "value": "E",
    "popup": {
      "menuitem": [
        {"name":"Undo", "value": "2U", "onclick": "undo()"},
        {"name":"Copy", "value": "2C", "onclick": "copy()"},
        {"name":"Cut", "value": "2T", "onclick": "cut()"}
        ]
      }


  }
]
```

Request Response Model

HTTP/HTTPS Request Response Communication



- In request/response communication mode.
- One software module sends a request to a second software module and waits for a response.
- The First software module performs the role of the client.
- The second, the role of the server,
- This is called client/server interaction.

Request Response Model

Application Level Client :

- HTTP Client is an application library used in client side application to generate request and receive response.
- HTTP Client's libraries varies from platform to platform.

Request Response Model

HTTP Request

HTTP Request is the first step to initiate web request/response communication. Every request is a combination of request header, body and request URL.

Http Request Segments

Request Area	Standard Data Type
Body	Simple String, JSON, Download, Redirect, XML
Header	Key Pair Value
URL Parameter	String

Request Response Model

HTTP Request Methods:

Method Name	Responsibilities
GET()	The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.
Head()	Same as GET, but transfers the status line and header section only.
POST()	A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
PUT()	Replaces all current representations of the target resource with the uploaded content.
DELETE()	Removes all current representations of the target resource given by a URI.

Request Response Model

Request Compare GET vs. POST:

Key Points	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	**Can be bookmarked**	Cannot be bookmarked
Cached	Can be	Never
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions

Request Response Model

HTTP Response:
Http response is the final step of request-response communication. Every response is a combination of response header, body an d cookies.

Http Response Segments:

Response Area	Standard Data Type
Body	Simple String, JSON, Download, Redirect, XML
Header	Key Pair Value
Cookies	Key Pair Value

Request Response Model

HTTP Response status messages

Code	Meaning	Description
200	OK	The request is OK (this is the standard response for successful HTTP requests)
201	Created	The request has been fulfilled, and a new resource is created
202	Accepted	The request has been accepted for processing, but the processing has not been completed
203	Non-Authoritative Information	The request has been successfully processed, but is returning information that may be from another source
204	No Content	The request has been successfully processed, but is not returning any content
205	Reset Content	The request has been successfully processed, but is not returning any content, and requires that the requester reset the document view

Request Response Model

HTTP Response status code

Code	Meaning	Description
206	Partial Content	The server is delivering only part of the resource due to a range header sent by the client
400	Bad Request	The request cannot be fulfilled due to bad syntax
401	Unauthorized	The request was a legal request, but the server is refusing to respond to it.
403	Forbidden	The request was a legal request, but the server is refusing to respond to it
404	Not Found	The requested page could not be found but may be available again in the future
405	Method Not Allowed	A request was made of a page using a request method not supported by that page

Request Response Model

HTTP Response status Code

Code	Meaning	Description
408	Request Timeout	Request Timeout
500	Internal Server Error	A generic error message, given when no more specific message is suitable
502	Bad Gateway	The server was acting as a gateway or proxy and received an invalid response from the upstream server
503	Service Unavailable	The server is currently unavailable (overloaded or down)

Request Response Model

Response Header:

- Provide proper http response status code.
- Provide proper content type, file type if any.
- Provide cache status if any.
- Authentication token should provide via response header.
- Only string data is allowed for response header.
- Provide content length if any.
- Provide response date and time.
- Follow request-response model described before.

Request Response Model

Response Body:

- Avoid providing response status, code, message via response body
- Use JSON best practices for JSON response body.
- For single result, can use String, Boolean directly.
- Provide proper JSON encode-decode before writing JSON Body.
- Follow discussion on JSON described before.

Request Response Model

Response Cookies:

- A Restful API may send cookies just like a regular Web Application that serves HTML
- Avoid using response cookies as it is violate stateless principle.
- If required use cookie encryption, decryption and other policies

Request Response Model

When use GET():

- GET is used to request something from server with less amount of data to pass.
- When nothing should change on the server because of your action.
- When request only retrieves data from a web server by specifying parameters
- Get method only carries request url & header not request body.

When use POST():

- POST should be used when the server state changes due to that action.
- When request needs its body, to pass large amount of data.
- When want to upload documents , images , video from client to server

Request Response Model

Request Body:

- Request body should be structured in JSON Array/ Object pattern
- Request body hold multipart/ form-data like images, audio, video etc
- Request body should not hold any auth related information.
- Request body should associated with specific request data model, setter getter can used for this

Request Header:

- Request header should carry all security related information, like token, auth etc.
- Only string **Key:Pair** value is allowed for header .
- Request header should provide user agent information of client application.
- If necessary CSRF/ XSRF should provide via header.
- Request header should associated with middleware controller, where necessary

Authentication

Bearer Authentication

গাড়ির গ্যারেজ এর টোকেন সিস্টেম

Authentication

JWT (JSON WEB TOKEN):

- Compact and self-contained way for securely transmitting information between parties as a JSON object.
- Information can be verified and trusted because it is digitally signed.

USES:

Authorization: Allowing the user to access routes, services, and resources

Information Exchange: Way of securely transmitting information between parties.