REST API DOCUMENTATION – READY, STEADY, GO!

APARNA MUDALIAR

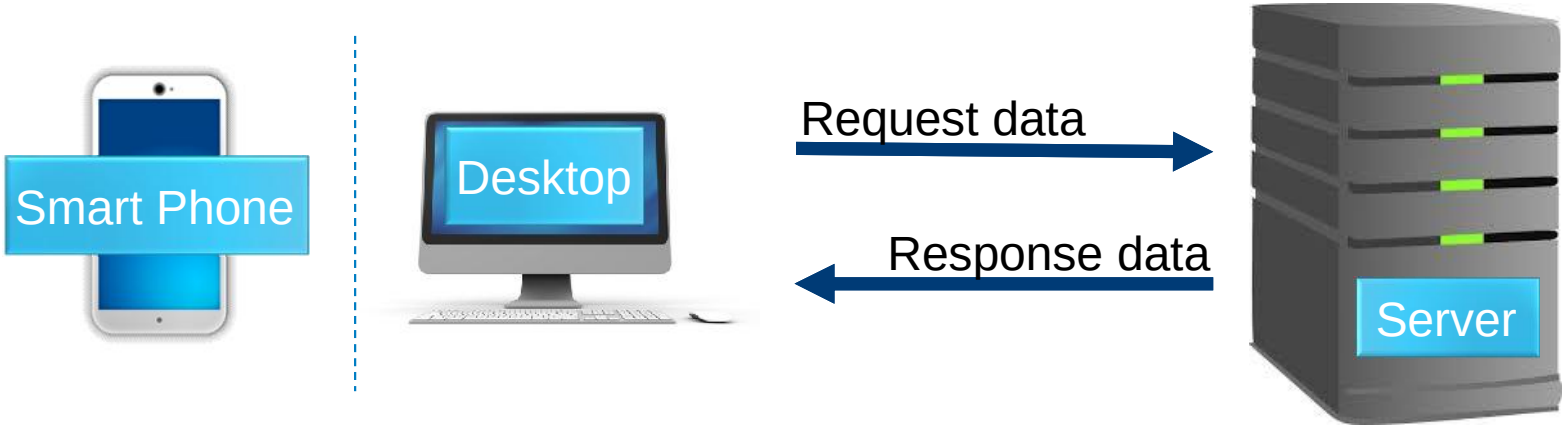
AGENDA

TOPICS

- Introduction to REST APIs
- Anatomy of a REST API
- Introduction to JSON
- Sample Template for Reference Documentation
- References
- Q & A

REST API | HOW DOES IT WORK?

- Full Forms
  - Representational State Transfer
  - Application Programming Interface
- Client-server interaction
- Exchange of data

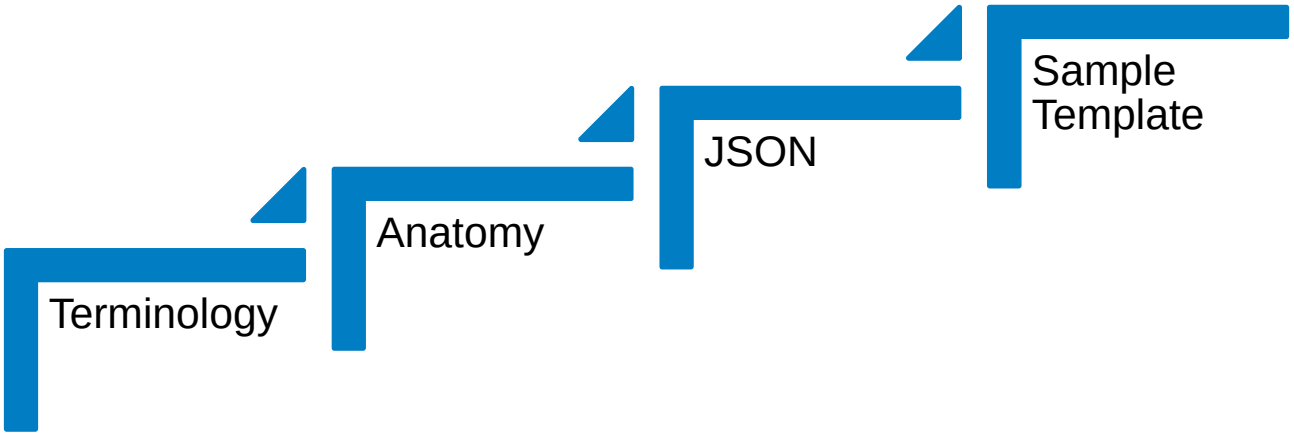


WHY IS REFERENCE DOCUMENTATION IMPORTANT?

- REST APIs are gaining popularity with developers
- Ultimate goal is to make the REST APIs reusable
- Reference documentation for developer audience
- Less support calls

REFERENCE DOCUMENTATION

STANDARD STEPS



REST API | TERMINOLOGY

- **HTTP Method**
  - CRUD (Create-Retrieve-Update-Delete) methodology
  - POST-GET-PUT-DELETE methods
- **Base URL**
  - URL of the server that receives the request and sends the response
- **Resources**
  - Real-world domain objects on which the REST API acts upon
- **Query Parameters**
  - Filter response data
- **Request URL**
  - HTTP method + Base URL + Resource representation + Query parameters

REST API | TERMINOLOGY

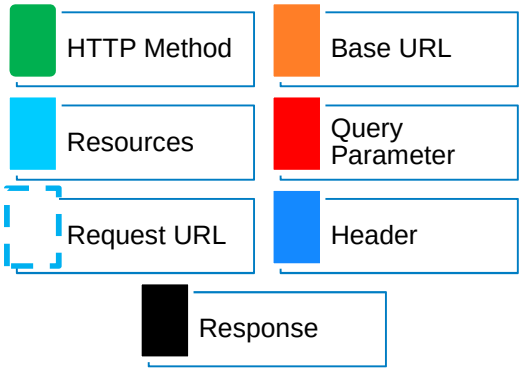
- **Header**
  - Defines the data format of requests and responses
  - Access token
- **Request body**
  - Request data sent to the server
  - Format – JSON, XML, Media
- **Response body**
  - Response data received from the server
  - Format – JSON, XML, Media
- **Response codes**
  - Indicate the success or failure of requests
  - Success codes
  - Failure codes

GET https://flickr.api.com/user/{userId}/photo?dateUploaded=2015-04-20

Bearer: 234c23488e8f

Accept: application/json

```
{   “photoFiles”:
    { “photoID”: “12345”,
      “url”: “https://flickr.api.com/user/photo/12345.jpg”,
      “height”: 2234,
      “width”: 1873
    }
}
```



JSON | INTRODUCTION

- Stands for JavaScript Object Notation
- Type of structured data format
- Requests and responses are in JSON format
- More popular than XML
  - Simpler notations
  - Lesser text
  - No markups

JSON | DATA TYPES

Collections

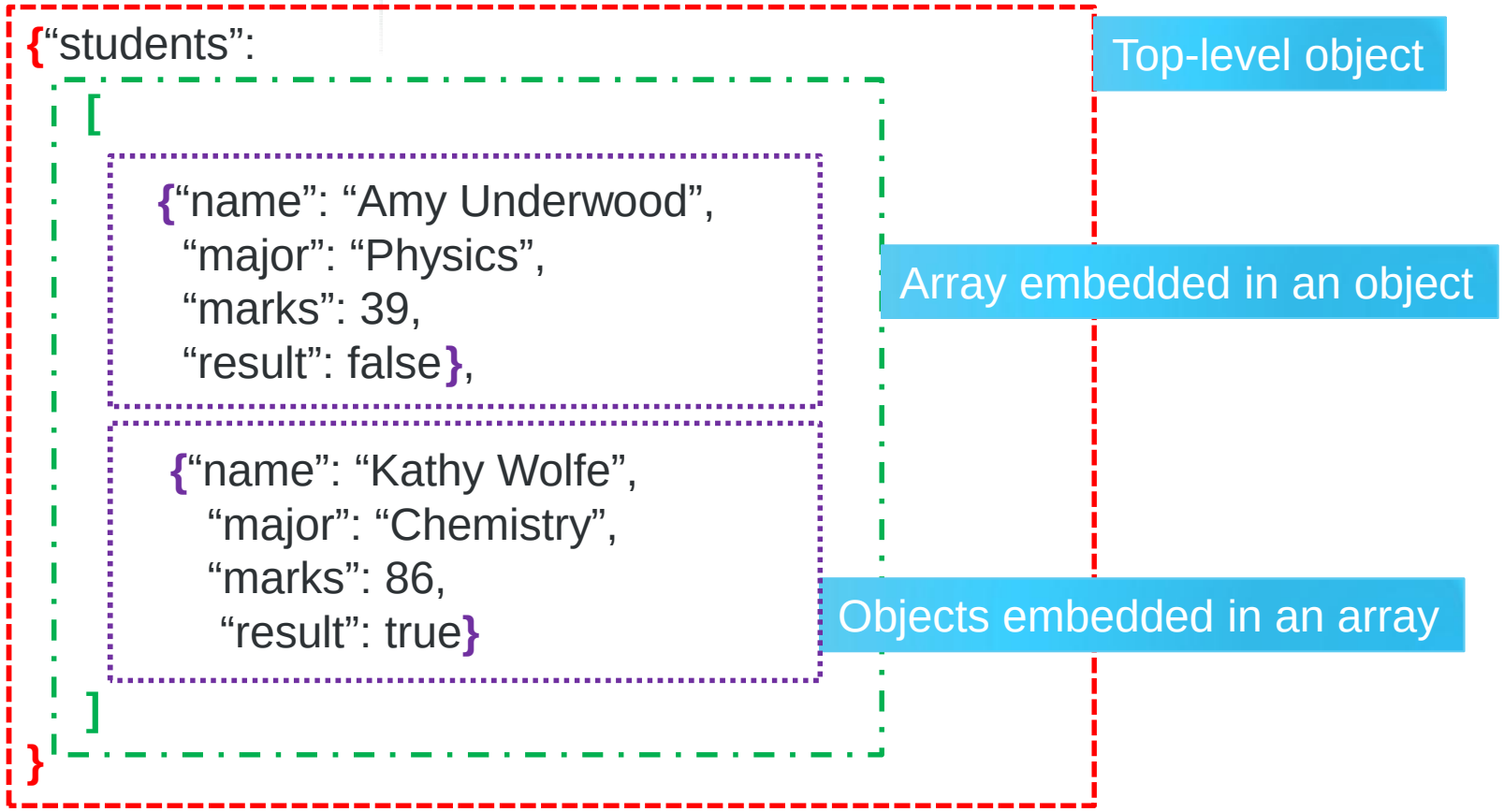
- **Arrays**
  - List of values
  - Comma separated
  - Enclosed in [ ]
- **Objects**
  - Key-value pairs
  - Key and value are colon (:) separated
  - Pairs are comma separated
  - Enclosed in { }

Values

- Number (Positive, Negative, Integers, Decimals)
- String (Enclosed in single or double quotes)
- Boolean (true or false)
- null (No quotation marks)

A JSON response or request is a single object with other objects and arrays nested in it.

JSON | SAMPLE RESPONSE



XML response

```
<students>
    <data>
        <name>Amy Underwood</name>
        <major>Physics</major>
        <marks>39</marks>
        <result>false</result>
    </data>
    <data>
        <name>Kathy Wolfe</name>
        <major>Chemistry</major>
        <marks>86</marks>
        <result>true</result>
    </data>
</students>
```

JSON response

```
{"students":
    [
        {"name": "Amy Underwood",
         "major": "Physics",
         "marks": 39,
         "result": false},
        {"name": "Kathy Wolfe",
          "major": "Chemistry",
          "marks": 86,
          "result": true}
    ]
}
```

EXAMPLE

REST API for Adding a Comment to a Photo

INPUTS FROM THE DEVELOPER FOR THE REST API

- Adds a comment to a photo as the currently authenticated user
- Uses the HTTP POST method
- URL is https://flickr.api.com/user/<userID>/photo/comment
- Header contains three elements: Bearer, Content-Type, and Accept (For more information, see the sample Header.)
- Request body contains two mandatory elements: photoID and commentText. (For more information, see the sample request)
- Response contains three elements for the comment object: id, authorName, and dateCreated. Additional status element. (For more information, see the sample response.
- Error messages: 400: Blank comment, 401: User not logged in, 404: Photo not found

EXAMPLE

ADDING A COMMENT TO A PHOTO

Click the sample template to view its content.

Sample Template: Adding a Comment

TABLE DOCUMENTATION

ADVANTAGES AND DISADVANTAGES

- Advantages
  - Simplistic approach
  - No special tools required
- Disadvantages
  - Formatting is difficult if nesting is too deep
  - Static documentation

Way ahead:

- Explore tools for interactive documentation
- Swagger (Example: http://petstore.swagger.io/)

REFERENCES | USEFUL RESOURCES

- Udemy Courses (by Peter Gruenbaum)
  - Learn API Technical Writing: JSON and XML for Writers
  - Learn API Technical Writing 2: REST for Writers
- REST API concepts and examples (YouTube video)
- I'd Rather Be Writing (Tom Johnson)

THANK YOU!