Understanding APIs and RESTful Architecture

1.4 Request / Response Lifecycle and Demo

Learning objectives

- What happens when a client calls an API
- Structure of an HTTP request
- Structure of an HTTP response
- Role of headers, parameters, and body
- Media types: JSON vs XML

The Client–Server Conversation

- Client sends a request to the server
- Server processes the request and sends back a response
- Communication happens over HTTP
- Every call follows this request → response pattern
- APIs exchange structured data, not HTML pages

Anatomy of an HTTP Request

Part	Description	Example
Method	Action to perform	GET, POST, etc.
URL	Address of the resource	/api/books/1
Headers	Metadata about the request	Content-Type: application/json
Body	Optional data sent to the server	JSON with user details

Anatomy of an HTTP Response

Part	Description	Example
Status Code	Outcome of the request	200 OK, 404 Not Found
Headers	Metadata about the response	Content-Type: application/json
Body	Data returned by the server	JSON with book info

Visualizing the Request–Response Flow

Understanding Headers

- Headers are key-value pairs with extra information
- Common examples:
  - Content-Type: Format of data being sent (application/json)
  - Accept: Format client expects in response
  - Authorization: Access token or credentials
- Important for API compatibility and security

Query Parameters

- Used to filter or search data in a request
- Placed after ? in the URL
- Example: GET /api/books?author=King&year=1985
- Often used with GET requests
- Not part of the request body

Request Body (Payload)

- Carries data to the server
- Used with methods like POST, PUT, or PATCH
- Commonly formatted as JSON
- Example: {"title": "Clean Code","author": "Robert Martin"}
- The server parses this data and performs an action

Media Types and Content Negotiation

- APIs use media types to describe data formats
- Defined in the Content-Type and Accept headers
- Common types:
  - application/json – most widely used today
  - application/xml – older, but still found in legacy systems
- JSON is preferred for REST APIs because it's:
  - Lightweight, Easy to read, Supported by most languages

JSON vs XML Example

- Key Difference: JSON uses key-value pairs and arrays; XML uses nested tags.

Json:

```
{
  "id": 1,
  "title": "The Pragmatic Programmer"
}
```

XML:

```
<book>
  <id>1</id>
  <title>The Pragmatic Programmer</title>
</book>
```

What Is JSON?

- JSON = JavaScript Object Notation
- A lightweight data format for storing and exchanging data
- Text based – easy to read and write
- Language independent but follows JavaScript syntax rules
- The default data format for most REST APIs

Why APIs Use JSON

- Simpler and smaller than XML
- Easy to parse and generate in any language
- Human-readable
- Excellent fit with object-oriented data models
- Works naturally with Java objects (later via Jackson)

JSON Basics: Objects

- Object = unordered set of key-value pairs
- Written inside curly braces { }
- Example: {"id": 1,"title": "Write slides","completed": false}
- Values can be string, number, boolean, array, or object

JSON Basics: Arrays

- Array = ordered list of values
- Written inside square brackets [ ]
- Example:[{ "id": 1, "title": "Task 1" },{ "id": 2, "title": "Task 2" }]
- Arrays can contain objects or primitive values

Data Types in JSON

Type	Example	Notes
String	"OpenAI"	Text enclosed in quotes
Number	42	Integers or decimals
Boolean	true	Only true or false
Null	null	Represents “no value”
Object	{…}	Key-value pairs
Array	[ ]	Ordered list of values

Demo 1: Viewing a Request & Response

- Goal: Show the full lifecycle of a REST API call.
- Access: https://jsonplaceholder.typicode.com/posts/1
- Show in browser: response in JSON → represents a single post.
- Open browser dev tools (Network tab) → refresh → click on the request.
- Explain: Request URL and Method (GET), Status code (200), Headers (e.g., Content-Type), Body (JSON data)
- Optionally show what happens if you try:
  - https://jsonplaceholder.typicode.com/posts/abc → 404 Not Found

Demo 2: Exploring a JSON Response

- Goal: See how real API responses look.
- Instructor Steps:
  - 1. Open browser and navigate to: https://jsonplaceholder.typicode.com/posts/1
  - 2. Show the raw JSON output.
  - 3. Identify: keys, values, and their data types.
  - 4. Then open https://jsonplaceholder.typicode.com/posts to show an array of objects.
  - 5. Explain the structure ([] for list, {} for object).

Lab 1: Analyze an API Request and Response

- Goal: Understand and describe what happens when a client makes an HTTP request.
- 1. In your browser, visit one or more of these APIs:
  - https://jsonplaceholder.typicode.com/users/1
  - https://api.agify.io?name=sarah
  - https://api.genderize.io?name=alex
- 2. Open browser Developer Tools → Network tab.
- 3. Refresh the page and select the API request.
- 4. Observe and note: The Request URL, The HTTP Method, The Status Code, The Response Body, The Content-Type header

Lab 2: Reading and Understanding JSON Data

- Goal: Practice recognizing objects and arrays in real API responses.
  - 1. Open your browser.
  - 2. Visit these URLs one by one:
    - https://jsonplaceholder.typicode.com/users
    - https://jsonplaceholder.typicode.com/posts/1
    - https://jsonplaceholder.typicode.com/comments?postId=1
  - 3. For each, answer these questions:
    - Is the top-level element an object or an array?
    - What keys and values do you see?
    - What data types are used (string, number, boolean)?
    - How many levels of nesting do you find?

Lab - Reflection

- What information can we find in headers?
- Why are status codes helpful for debugging?
- How does the response format affect API clients?
- Did all APIs return JSON?

Key Takeaways

- Every API call follows a request → response flow
- Requests include method, URL, headers, and body
- Responses include status, headers, and body
- Headers describe data format and metadata
- JSON is the standard format for REST APIs
- Understanding this flow is essential for building and testing APIs