01 A Look Under the Hood at REST APIs

A Look Under the Hood at REST APIs

Introduction to Web Services

A **web service** is a defined interface that allows two systems to communicate over a network. **REST (Representational State Transfer)** APIs are a popular type of web service that leverage the HTTP protocol to facilitate communication between systems. They are widely used due to their simplicity, scalability, and alignment with web standards. This lecture provides a comprehensive overview of REST APIs, their structure, and practical applications.

What is a REST API?

REST APIs operate on the **HTTP request and response model**, typically exchanging data in **JSON** format (though XML or other formats may be used). They are designed around the **CRUD** model, which stands for **Create, Read, Update, Delete**, mapping to specific HTTP verbs:

HTTP Verb	CRUD Action	Description
POST	Create	Creates a new resource
GET	Read	Retrieves a resource or list of resources
PUT	Update	Replaces an existing resource with new data
PATCH	Update	Partially updates an existing resource
DELETE	Delete	Removes a resource

To effectively use a REST API, consult its **documentation**, which details supported HTTP verbs, endpoints, parameters, authentication requirements, and expected responses.

Anatomy of a REST API URI

REST APIs typically start with a URI. You can study the parts of this URI to determine what you are requesting.

A REST API request begins with a **Uniform Resource Identifier (URI)**, which specifies the resource to interact with. Consider an example from the Google Maps API:

```
https://maps.googleapis.com/maps/api/geocode/json?address=sanjose
```

URI Components

1. **Protocol:**
   - `http://` or `https://` indicates whether the connection is unsecured or secured (encrypted).

- Example: `https://` ensures secure communication.

2. **Domain Name**:
   - `maps.googleapis.com` is the server or host providing the service.
   - May include a port number (e.g., `:8080`) for non-standard ports.
3. **Resource Path**:
   - `/maps/api/geocode/json` identifies the specific endpoint or resource.
   - Typically uses plural nouns (e.g., `/users`, `/facts`) and descriptive words.
4. **Parameters**:
   - `?address=sanjose` provides optional query parameters to filter, scope, or paginate the response.
   - Multiple parameters are separated by `&` (e.g., `?key=value1&param2=value2`).

Understanding these components is essential for constructing valid API requests. Always refer to the API's documentation to identify the correct URI structure and available parameters.

Practical Example: Random Useless Facts API

Let's explore the Useless Facts API to demonstrate REST API concepts. This API provides random or daily facts in JSON format, with support for English (`en`) or German (`de`) languages.

Fetching a Random Fact

Use the `curl` command to send a **GET** request to the random fact endpoint:

```
curl https://uselessfacts.jsph.pl/api/v2/facts/random | python -m json.tool
```

This command retrieves a random fact and uses Python's `json.tool` module to format the JSON response for readability. Example output:

```
{
  "id": "12345",
  "text": "The shortest war in history lasted 38 minutes.",
  "language": "en",
  "source": "some_source"
}
```

Adding a Query Parameter

To request a fact in a specific language, append the `language` parameter:

```
curl https://uselessfacts.jsph.pl/api/v2/facts/random?language=en | python -m
json.tool
```

This ensures the response is in English. For German, use `?language=de`.

Fetching the Fact of the Day

The API also provides a "fact of the day" endpoint, which returns the same fact for 24 hours:

```
curl https://uselessfacts.jsph.pl/api/v2/facts/today?language=en | python -m
json.tool
```

Python Example: Fetching and Displaying a Fact

Below is a Python script to programmatically fetch and display a random fact or the fact of the day, with error handling:

```
import requests
import json


def fetch_fact(endpoint="random", language="en"):
    """
    Fetch a fact from the Useless Facts API.


    Args:
        endpoint (str): API endpoint ('random' or 'today')
        language (str): Language code ('en' or 'de')
    """
    url = f"https://uselessfacts.jsph.pl/api/v2/facts/{endpoint}?language=
{language}"
    try:
        response = requests.get(url)
        response.raise_for_status()  # Raise an error for bad status codes
        fact = response.json()
        print(f"Fact: {fact['text']}")
        print(f"Language: {fact['language']}")
        print(f"Source: {fact['source']}")
    except requests.exceptions.RequestException as e:
        print(f"Error fetching fact: {e}")

if __name__ == "__main__":
    print("Fetching a random fact:")
    fetch_fact("random", "en")
    print("\nFetching the fact of the day:")
    fetch_fact("today", "en")
```

Run this script to fetch and display a random fact in English. Modify the language parameter to "de" for German.

Response Status Codes

When you send a request, the API responds with a **status code** indicating the result. Common status codes include:

Status Code	Status Message	Meaning
200	OK	Request successful
201	Created	New resource created
400	Bad Request	Invalid request
401	Unauthorized	Missing or incorrect authentication
403	Forbidden	Request understood but not allowed
404	Not Found	Resource not found

Status Code	Status Message	Meaning
500	Internal Server Error	Server-side error
503	Service Unavailable	Server unable to complete request

To view the status code and headers, use the -i (--include) option with curl:

```
curl --include https://uselessfacts.jsph.pl/api/v2/facts/random
```

Example output:

```
HTTP/1.1 200 OK
Access-Control-Allow-Headers: Accept, Content-Type, Content-Length, Accept-
Encoding, Authorization
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
Access-Control-Allow-Origin: *
Content-Length: 280
Content-Type: application/json; charset=UTF-8
Date: Tue, 19 Aug 2025 07:27:02 GMT
Server: nginx

{"id":"76d9063d8bc889cc22659c46a05ac562","text":"PEZ candy even comes in a Coffee
flavor.","source":"djtech.net","source_url":"http://www.djtech.net/humor/useless_f
acts.htm","language":"en","permalink":"https://uselessfacts.jsph.pl/api/v2/facts/7
6d9063d8bc889cc22659c46a05ac562"}
```

The first line (HTTP/1.1 200) confirms a successful request.

Authentication in REST APIs

Many REST APIs require **authentication** to verify the client's identity. Common methods include:

- **Username/Password**: Generates an expirable token.
- **OAuth2**: Integrates with an identity provider (e.g., Cisco Webex APIs).
- **API Key**: A unique key for programmatic access (e.g., Cisco Meraki).

Check the API documentation for authentication requirements. The Useless Facts API, for example, does not require authentication.

REST API Headers

Headers provide metadata for requests and responses. Common headers include:

Header	Example Value	Purpose
Content-Type	application/json	Specifies the format of the request body
Accept	application/json	Specifies the desired response format
Authorization	Bearer aBcDeFgHiJklMnOor0pQrS5z	Provides credentials (e.g., token)
Date	Thu Jan 4 17:02:23 CST 2018	Timestamp of the request or response

Headers are critical for authentication, data formatting, and other metadata needs.

Key Takeaways

- REST APIs use HTTP verbs (GET, POST, PUT, PATCH, DELETE) to perform CRUD operations.

- URIs consist of protocol, domain, resource path, and optional parameters.
- Status codes (e.g., 200 OK, 404 Not Found) indicate the result of a request.
- Authentication varies by API, ranging from none to OAuth2 or API keys.
- Headers provide metadata, such as content type or authorization tokens.
- Always consult the API documentation for specific details on endpoints, parameters, and authentication.

Hands-On Exercise

1. Run the provided Python script to fetch a random fact.
2. Modify the script to fetch the fact of the day (/today endpoint).
3. Experiment with the language parameter (en or de).
4. Use curl with the --include option to inspect response headers.

Additional Resources

- **Useless Facts API Documentation:** https://uselessfacts.jsph.pl
- **Requests Library (Python):** https://requests.readthedocs.io
- **HTTP Status Codes:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Status