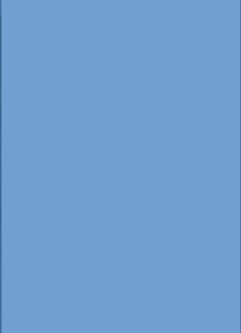




DOCUMENTING REST APIS

MAY 4, 2017

Tara English-Sweeney, Senior Information
Developer



In this session

- **What is an API?**
- **What is REST?**
- **Query Parameters**
- **Authentication and Authorization**
- **Request Bodies and Responses**
- **Errors**

What is an API?

- Application Programming Interface
 - It is a way to share data across the internet
 - Most people are referring to REST APIs today
 - Target audience is software developers



What is REST?

- REpresentational State Transfer
- A design pattern, not a protocol
- Uses HTTP to send requests and responses, can use HTTPS for security
- REST APIs send data to a URL
- Data can be any format – JSON, XML, media, and so on
- Uses verbs such as GET, POST, PUT, and DELETE
- SOAP is Simple Object Access Protocol, supports XML only and is not so simple

OAuth^{PUT}
JSON^{GET}_{POST} **XML**
REST
Basic-Auth
DELETE

Requests

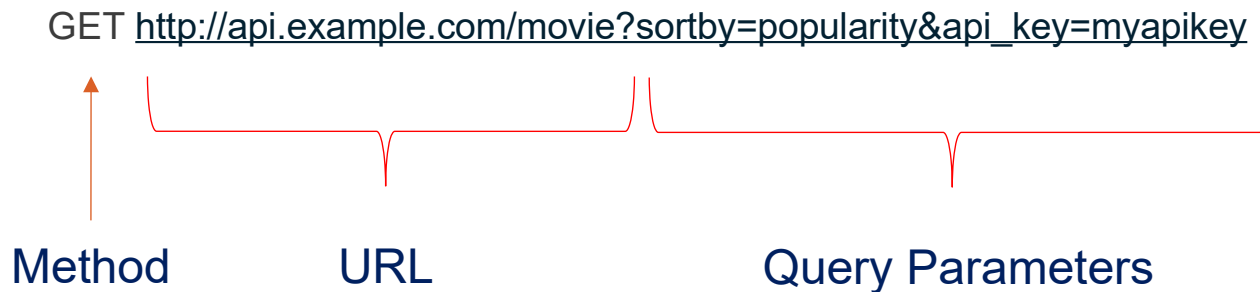
- Requests are sent from the device to the server
- The data sent to the server is packaged in an HTTP request
- The data sent back is called the *response*



Requests

- Method – Verb that describes the action (GET, POST, etc.)
- URL – What you take the action on (contains the domain and resource)
- Query Parameters – Refines the action (sorting, authorization information)

Simple request:



This request lets the server know that this request is allowed, and then retrieves a list of movies and sorts them in order of popularity.

Requests

Requests can have more information such as:

- Headers – Store information about the request
- Body – Contains additional data

Sample request:

```
POST http://api.example.com/user?source=ios&device=ipad  
Accept: application/json  
Content-type: application:json  
{  
  "name": "Taryn Jones"  
  "email": tjones@example.com  
}
```

Header

Body

This request created a user with the name Taryn Jones with the email tjones@example.com on an iPad.

Methods

- Represent the action you want to take with the request
- GET returns data from the server
- POST creates new data on the server
- DELETE deletes data from the server
- PUT updates data on the server



URL

- Starts with http or https for secure transactions
- Contains the domain such as example.com
- Includes the resource
 - A resource is a piece of data that represents something
 - If you are making a request about movies, the URL should have the word movies in it – <http://api.example.com/movies>
 - You can add an ID to get information about a specific resources – <http://api.example.com/movies/12345>

HTTP methods / URIs for collection/item

	http://api.co/v2/cars/	http://api.co/v2/cars/1234
GET	List all the cars	Retrieve an individual car
POST	Create a new car	Error
PUT	Replace the entire collection with a whole new list of cars	Replace or create an individual car
DELETE	Delete all the cars	Delete an individual car

Query Parameters

- Way for the API to provide additional information to the server to modify or filter the data that is returned
- Typically part of the URL
- Comes in key value pairs after the ?
- Use an ampersand (&) between them

<http://api.example.com/movies?genre=action&sortBy=popular>

- Can be used for authorization

<http://api.example.com/movies?token=3f8gw0sfjlwsrf9wertlk>



Documenting Query Parameters

- Document the following:
 - Parameter
 - Description
 - Type (Integer, Date, String, etc.)
 - Required
 - Notes

GET <http://api.example.com/weather?date=2017-22-04&days=10&zip=10009&language=en>

Parameter	Description	Type	Required	Notes
Date	First day of the forecast.	Date	Optional	Format is YYYY-MM-DD. Default is today's date.
Days	Number of days of the forecast.	Integer	Required	
Zip	US postal code for the location of the forecast.	Integer	Required	
Language	Language for returned data.	String	Optional	Two letter language format. Valid values are "en" (default), "fr", and "es".

Headers

- Part of the HTTP protocol
- Can be used to specify formats for sending and receiving data
- Most commonly used for data formats and authorization
- Examples of most used headers for formatting
 - Accept – formats accepted by the client (Accept: application/json)
 - Content-Type – format associated with the request body (Content-Type: image/jpeg)

POST <http://api.example.com/photo>

Content-Type: image/jpeg

Accept: application/json

Header	Description	Required	Values
Content-Type	Specifies the format for the photo to upload.	Optional	Image/jpeg (default), image/png, image, gif
Accept	Specifies the format for the data returned.	Optional	Application/json (default), application/xml

Authentication and Authorization

- Only registered developers should have access to call APIs
- User information should remain private
- Authentication
 - User supplies a valid name and password to receive an access token
- Authorization
 - When you make an API request, you send an access token, which is like a key



OAuth

- A popular open protocol that is both complex and flexible
- Access tokens tell the server what the API user can access
- There are different grant types and different types of authorization
 - One-legged authorization
 - Used if there is no sensitive data involved, such as user data. For example, viewing LinkedIn profiles.
 - Three-legged authorization
 - Used when you need to protect user data. For example, accessing a LinkedIn user's connections.



Authentication and Authorization

- You will need to talk to your development team to document authorization
- Include the following in documentation
 - The URL of the authentication server
 - Step by step about how the process works to get an access token
 - How to get the access token from the server's redirect URL
 - How to pass a token to the server
 - When a token will expire and how to get a new one
 - How is the token passed for each API request – through a header or a query parameter

Request Bodies and Responses

- Only POST and PUT methods requests have bodies, sometimes DELETE
- Typically use json or xml
- Sometimes many API requests return the same type of data and the response and sample response are redundant.
 - If so, create an Objects section in your document with the Response Body tables
 - From the Request body section, link to the Object section
- To document responses, include:
 - Element
 - Description
 - Type
 - Notes
 - Required (only for requests)

Request: /orders?start=2015-10-01T00%3A00%3A00Z&end=2015-10-04T00%3A00%3A00Z

Status: 200

Latency: 49 ms

▼ Response Body

```
[
  {
    "orderId": "order-0",
    "orderTs": "2015-10-01T00:00:00.000Z",
    "orderAmount": 47
  },
  {
    "orderId": "order-1",
    "orderTs": "2015-10-02T00:00:00.000Z",
    "orderAmount": 22
  },
  {
    "orderId": "order-2",
    "orderTs": "2015-10-03T00:00:00.000Z",
    "orderAmount": 52
  }
]
```


Errors

- Errors returned with a response use HTTP status codes
- Document the code, the standard description, and additional information

Code	Description	Notes
200	OK	Success
401	Unauthorized	The access token is not valid for this resource.
403	Forbidden	The number of API requests per month has been exceeded.
404	Not found	The resource name or ID is invalid.

Conceptual Material

- **API review**
- **Why write good API documentation?**
- **About API documentation**
- **Conceptual material**
- **Overview**
- **Getting started**
- **Tutorials**

Let's review what an API is...

- Defines how two pieces of software talk to each other
- Most common APIs are Web APIs such as REST, SOAP, etc.
 - Others are Platform (Java, C++, etc.)

Why write good API documentation?

- Increases development adoption of the API
- Decreases support costs
- Developers must quickly understand what the API can do
- Developers want to get up and running quickly
- Writers write instead of developers because:
 - Developers are too close to the code
 - Developers focus on the how and forget why



API Documentation

About API documentation

- Target audience is software developers
- Conceptual documentation
 - Focuses on high level information
 - Gets developers oriented and started using the API
 - Helps them with common tasks
- Reference documentation
 - More detailed
 - Details about HTTP Requests and Responses for Web APIs
 - Details about classes, methods, properties, etc. for Platform APIs

Conceptual material

- Overview
- Getting Started
- Tutorials
- Sample Code/Examples

Overview

- What can the API do
- Requirements for using the API
- Why to use the API
- Key concepts
- Workflow (order of steps for common tasks)
- Visual information (architecture or data models)

Getting started

- Why you need it
- Hello World
- Structure
 - Step-by-step instructions
 - For Web APIs
 - Registration
 - Get an app key
 - Making 1-2 HTTP requests that return a response
 - Doing something simple (typically not something that requires complex authorization)
 - Platform
 - Download SDK
 - Set up your IDE
 - Do something simple – provide code to be copied and pasted

Tutorials

- Include step by step instructions for several common tasks
 - Start with basic tasks and get more advanced
 - Examples are Getting account information, Adding a student to a course
- Same structure as used in the Getting Started section
 - Simple short programs that work
 - Step by step instructions
 - Sample HTTP requests or code that they can copy and paste
 - Screen shots if they are valuable
 - Web APIs – typically not needed unless doing something with the developer portal
 - Platform APIs – only if doing something with the IDE or if the tutorial does something visual
- Include 3-5 tutorials, unless it is a very complex API

Tools for REST APIs

- Get a better feel for how the REST API works
- Try out requests and answer questions without having to ask the dev team
 - Is a parameter required? Make request and see if you get an error
- GUI tools such as Postman
- Command line tools such as cURL