The Learning House

427 S 4th Street #300

Louisville KY 40202

Introduction to HTTP

.NET Cohort

Coding Bootcamp

SOFTWARE GUILD

Why Do We Care?

If you're building apps or websites today, the hot term to know is building a REST API over HTTP.

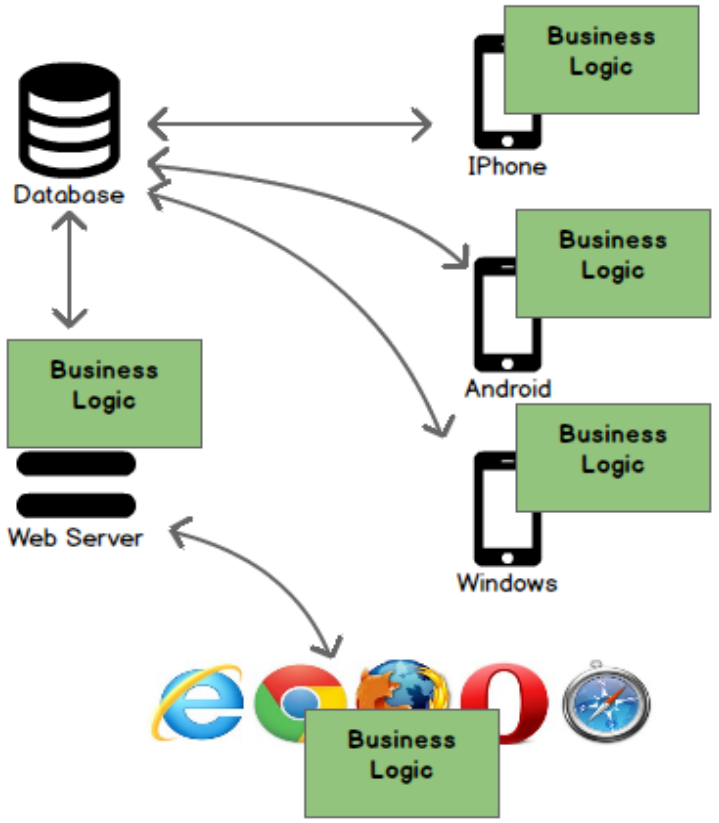
API stands for Application Programming Interface, which, in more human terms, refers to code intended to be shared among many applications to retrieve and update data.

Why Bother with an API?

Consider a modern application which has several mobile apps as well as a website (Evernote, Twitter, Facebook, etc.).

Without an API, you would repeat business logic in the applications.

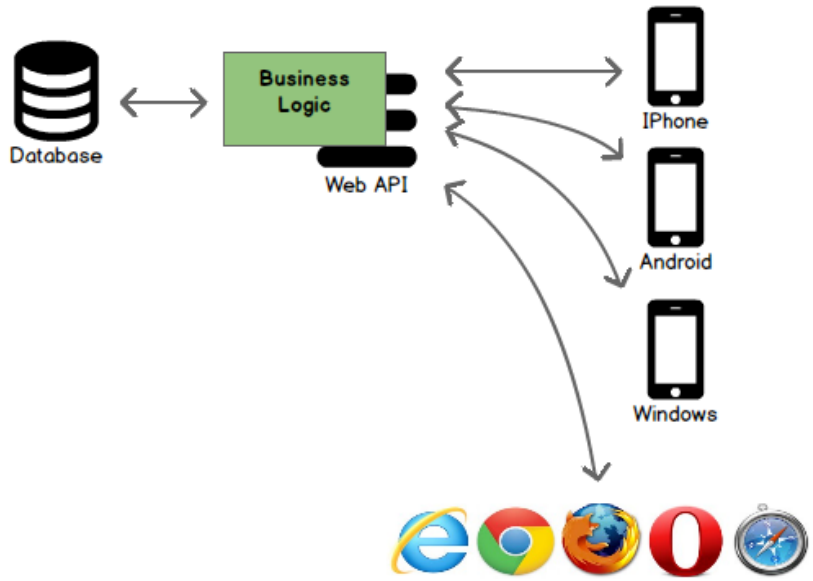
The big downside is that any changes to the database require all the business logic to be kept in sync.



Same App, With a Shared API

Consider the same application with the business logic exposed through a web API that holds all the business logic.

If each app uses the same API to manipulate data, all the apps can become lightweight UI layers instead of dealing with business logic and database structures.



About HTTP

Almost every device connected to the internet understands HTTP because it is the base protocol that the internet is built on.

This makes it a great platform for hosting an API.

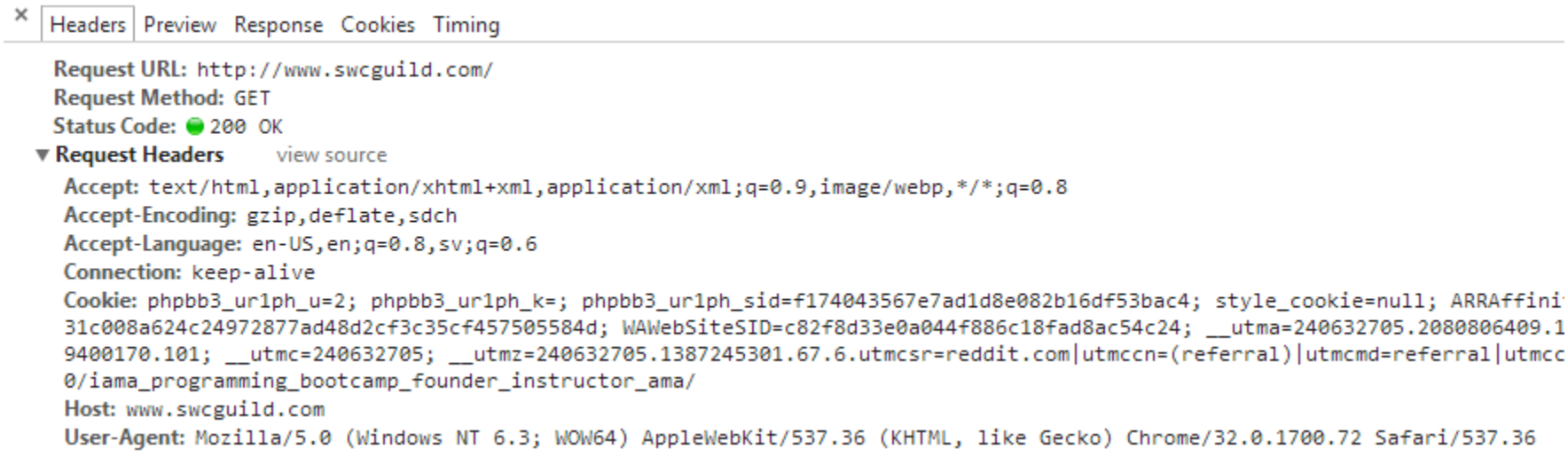
HTTP is a *request* and *response* system. The client sends a request to a server and the server sends back a response.

Key HTTP Terms We Need to Know

Term	Description
Resources	Resources can be many things: web pages, files, code, data services, etc. Resources are identified via URL.
Request Verbs	What do you want to do with the resource? GET, POST, PUT, DELETE, etc.
Request Headers	Additional instructions on the request, e.g. authorization
Request Body	Data being sent with the request, e.g. the form data a user filled out
Response Body	The main body of the response; can contain HTML, data, or even be a file download
Response Status Code	Tells the client the status of the request, e.g. 200: OK, 404: Not Found

Viewing HTTP Requests

In a browser, like Google Chrome, you can use the dev tools (F12) to view HTTP request header details. The response in this case is a web page (HTML).



SOFTWARE GUILD

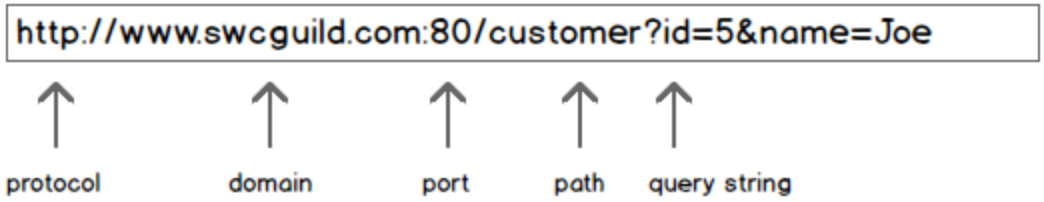
URLs

Uniform Resource Locators are paths to resources that you want to work with. URLs can return many types of resources:

- Web pages
- Downloadable files
- Data (XML, JSON, etc.)
- and more

URLs contain as much data as is needed to uniquely identify a resource.

URL Format



- Protocol (or scheme) tells the browser to use HTTP
- Domain (or IP address) is the destination server
- Port is optional; if omitted, the default for the protocol is used (80 for HTTP)
- Path helps specify which resource to use
- Query string is data that needs to be passed to the server

SOFTWARE GUILD

HTTP Verbs

HTTP verbs tell the server what to do with the request incoming from the client. The most common is GET.

- GET tells the server to transmit data identified by the URL.
- POST sends data to a server for processing. It is often used to create data.
- PUT can also be used to create, but it is more often used for updates.
- DELETE deletes a resource

Note that GET should not change server data, the other variables do change data, and some server designs use POST for everything.

Response and Status Codes

Every HTTP request response header has a status or response code that the server uses to tell the client what happened. Common codes:

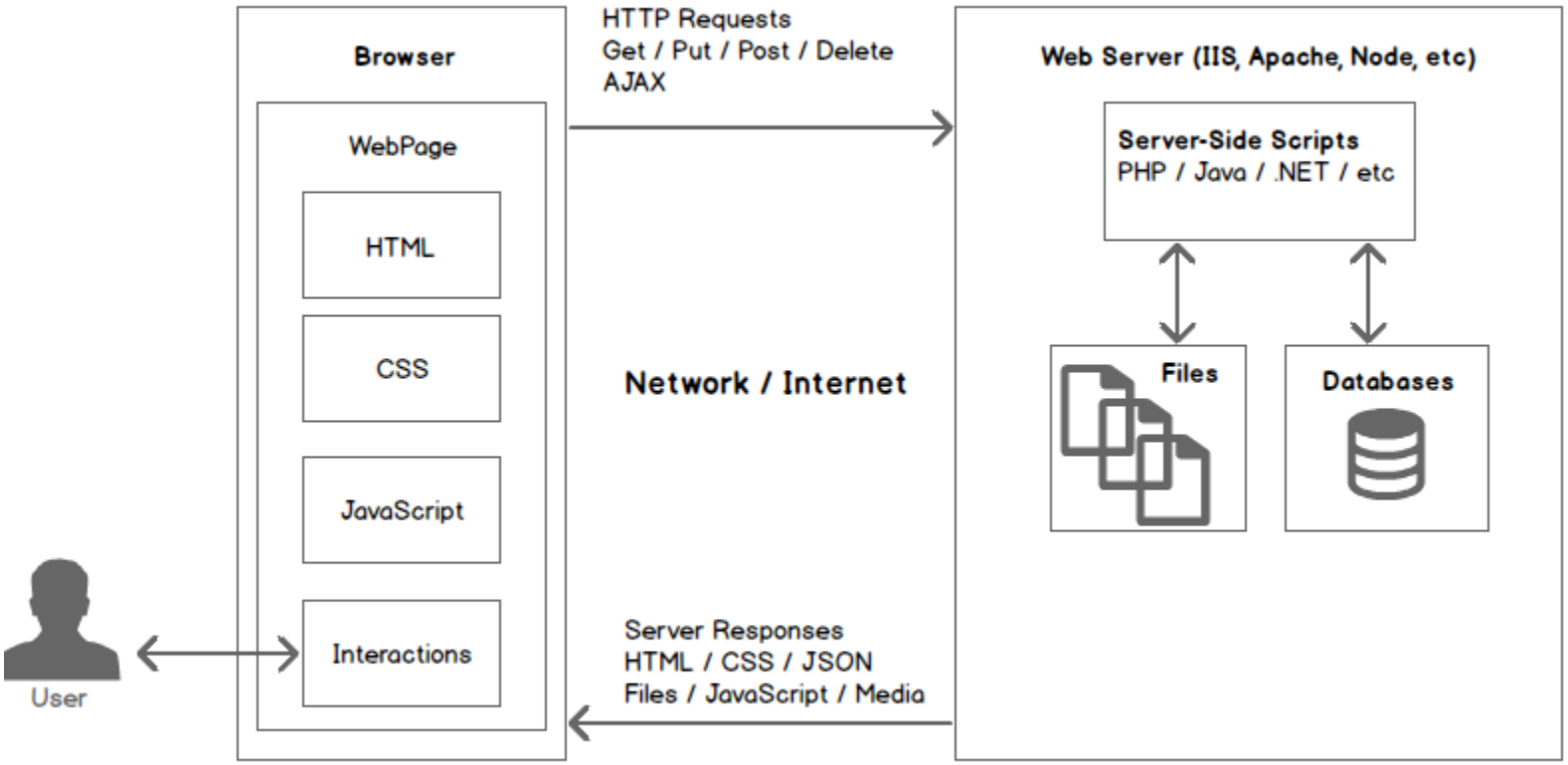
- 200 OK: the request was successful
- 201 CREATED: the request was success and a resource was created (POST/PUT)
- 400 BAD REQUEST: The request was malformed (bad data)
- 401 UNAUTHORIZED: Security doesn't allow you to access this resource
- 404 NOT FOUND: URL doesn't exist
- 405 METHOD NOT ALLOWED: The HTTP verb you used isn't supported
- 409 CONFLICT: A conflict such as putting the same resource twice.
- 500 INTERNAL SERVER ERROR: Processing failed (usually due to an exception occurring in the server code)

HTTP Status in a Nutshell

- 1xx: Hold on
- 2xx: Here you go
- 3xx: Go away
- 4xx: You messed up
- 5xx: I messed up

Full list (or Google “HTTP status codes”):
http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Request/Response Diagram



What is REST?

REST stands for Representational State Transfer and is an architecture pattern for HTTP APIs.

The original dissertation by Roy Fielding can be found here:

http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

REST APIs typically use data models (Product, Order, Customer) as resource names and use HTTP verbs to tell the server what to do with the resource (GET retrieves a customer, POST adds a new customer, etc.).

Some Typical REST Requests

Resource	Verb	Expected Outcome	Status Code
/Products	GET	List all products	200 OK
/Products?Category=sports	GET	List only products in the sports category	200 OK
/Products	POST	Create a new product	201 CREATED
/Products/81	GET	Get the product details for product id 81	200 OK
/Products/512	GET	In this example, imagine there is no product with id 512	404 Not Found
/Products/81	PUT	Update product information for product 81	204 No Content
/Products/81	DELETE	Delete product information for product 81	204 No Content

We Will Go Deeper Later

For now, just realize that the URL structure of websites, be they REST APIs or MVC web pages, is very important.

The URL plus the verb tells the server which methods to invoke and what kind of responses should be returned.

The request header and body provide additional information for processing the request.

The response header will always provide a status code and the body will sometimes return data to the client.

Gut Check

https://www.google.com/search?q=hello+world

- Which part is the…

Gut Check (2)

What are each of these requests doing, in plain English?

- GET /users/42
- PUT /users/83
- POST /users
- DELETE /users/13
- GET /users?name=Inigo+Montoya