Data Representation and Querying

ian.mcloughlin@gmit.ie

Topics

HTTP

Data

REST

Sessions

HTTP

HyperText Transfer Protocol

HyperText Text with links.

Transfer Communication of data.

Protocol Set of rules for communication.

HTTP/1.0 was the first version.

HTTP/1.1 is the current widely used version.

HTTP/2 use is slowly growing.

Why study HTTP?

HTTP is the main protocol used by web browsers.

Netflix uses HTTP to stream video (using DASH).

Instagram is basically just a HTTP API.

Facebook is a web application using HTTP.

GMail uses HTTP.

Twitter uses HTTP.

Medium is really clever about its use of HTTP.

Sandvine suggest most internet traffic happens over HTTP.

sandvine.com/downloads/general/global-internet-phenomena/2015

Why is HTTP so widely used?

HTTP is often used instead of protocols that are more suited to the application.

Browsers are one of the main reasons for this. Modern operating systems come with one or more browsers installed by default.

Web servers and browsers mainly talk over HTTP.

Libraries exist for most programming languages to make HTTP requests.

HTTP is relatively straght-forward.

Firewalls usually do not block HTTP by default.

What is HTTP?

You will know HTTP from typing `http://` in your browser's location bar.

RFC 2616 details HTTP/1.1.

HTTP is a standard way of transmitting data at the application level.

Originally it was for transmitting text.

Text is strings of ones and zeroes chopped into chunks, where each chuck represents a letter or character.

Nothing stops us from sending non-text data via HTTP.

How does HTTP work?

Clients performs requests. Firefox is an example of a client.

Servers respond to requests. Apache is an example of a server.

Requests are text documents sent by clients to servers.

Responses are text documents sent by server to clients.

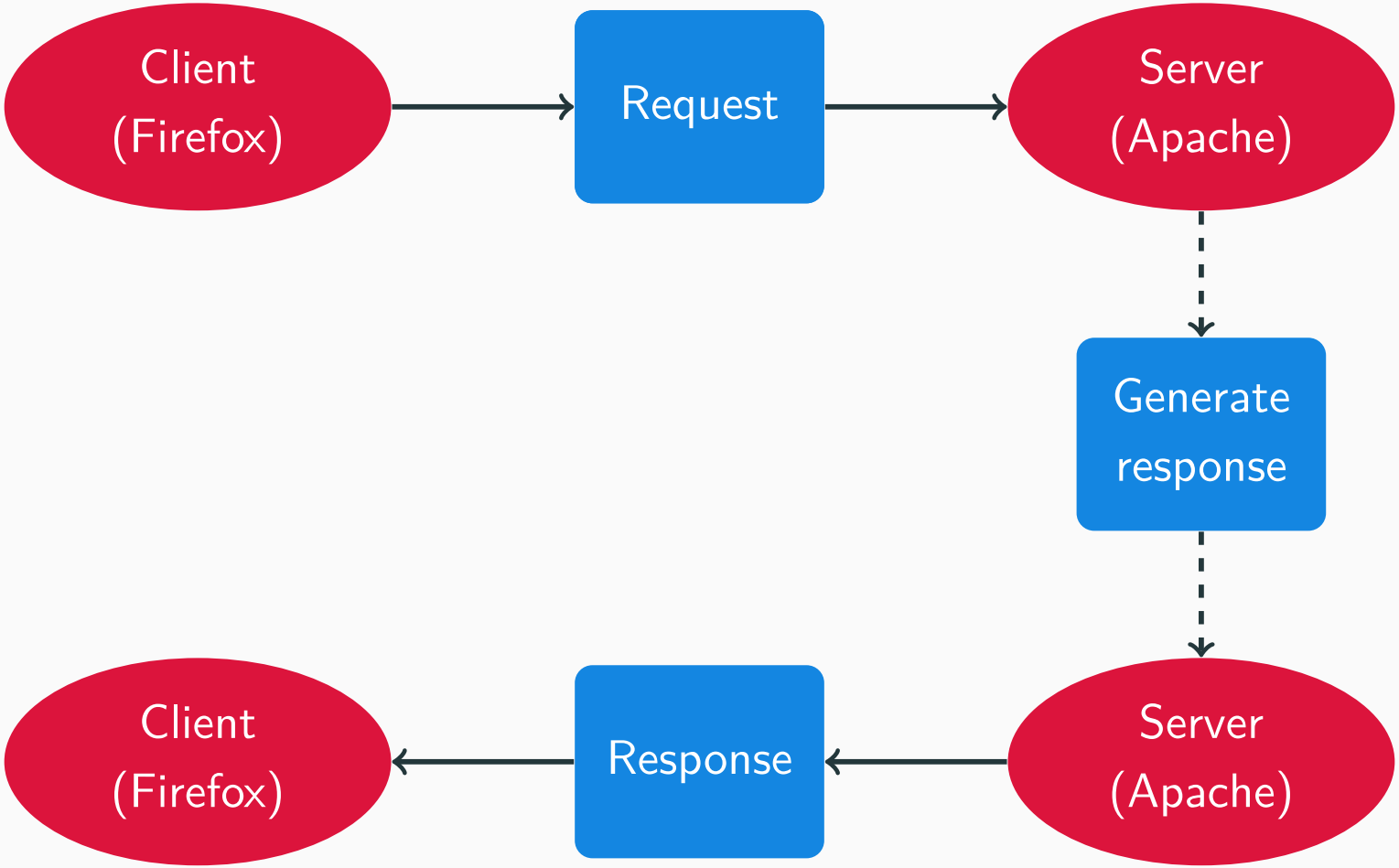
URLs specify the server's location and the resource on the server.

Headers are text metadata added to the start of requests and responses.

Bodies are the main content of responses, and sometimes requests.

HTML typically forms the body of a response.

Request–Response



HTTP is not like a phone call

Suppose I ring you and ask for your PPS Number.

If you don't understand the question, you can say "Sorry, can you please repeat that?".

Then I repeat my question, and you then give me the number before hanging up.

HTTP doesn't work like that.

Misunderstandings result in the server responding with an error, and hanging up.

Status codes indicate errors, amongst other things.

Uniform Resource Locator

http://username:password@www.reddit.com:80/r/funny/?limit=1

http	Protocol
username	Username
password	Password
www	Subdomain
reddit.com	Domain
80	Port
/r/funny/	Path
limit=1	Parameter

Request and Response Format

Requests and responses both have this format:

- Intial line.
- Zero or more header lines.
- A blank line.
- Optional message body (e.g. a HTML file)

Request (GET) Example

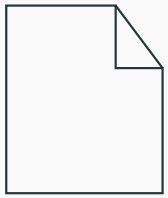
```
GET /courses/all-courses HTTP/1.1
Host: gmit.ie
User-Agent: curl/7.50.1
Accept: */*
```

Response Example

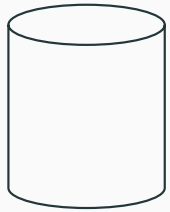
```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed

<html>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Resources



File



Database

HTTP is used to transmit resources . . . A resource is some chunk of information that can be identified by a URL . . . The most common kind of resource is a file, but a resource may also be a dynamically-generated query result . . .

HTTP Methods

GET	Retrieve information from the server.
HEAD	Like get, but retrieve only the response header.
POST	Send data to the server.
PUT	Set the resource at the URL to the request data.
DELETE	Delete the resource at the URL.
CONNECT	Set up tunnel for other traffic to pass through HTTP.
OPTIONS	Find the allowable operations at the given URL.
TRACE	Echo the received request.
PATCH	Partial resource modification.

Status codes

404 is probably the most famous status code. It means you requested a resource that doesn't exist on the server.

200 means everything is OK, and is the most common one in everyday browsing.

All status codes are three digit numbers.

1xx indicates an informational message only

2xx indicates success of some kind

3xx redirects the client to another URL

4xx indicates an error on the client's part

5xx indicates an error on the server's part

www.jmarshall.com/easy/http

Sending data to the server

Sometimes we want to tell the server something extra.

Resources can be generated differently based on this extra data.

Google can use this to perform searches on google.ie

Try opening `google.ie` in your browser, and then open `google.ie/?q=gmit`.

The requested resource here is the same but we don't get the same response.

Two main ways to send extra data to the server: in the URL after a question mark, or in the body of the request.

URL encoding

HTML form data is usually URL-encoded by changing:

- Unsafe characters to % *xx* where *xx* is the ASCII value.
- All spaces to plusses.
- Names and values to: name1=value1&name2=value2.

GET — in the URL after a question mark.

POST — in the body.

www.jmarshall.com/easy/http

Request (GET) with parameters

```
GET /r/ireland?limit=1 HTTP/1.1
Host: reddit.com
User-Agent: curl/7.50.1
Accept: */*
```

Request (POST) Example

```
POST /path/script.cgi HTTP/1.0
From: frog@jmarshall.com
User-Agent: HTTPTool/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 32

home=Cosby&favorite+flavor=flies
```

Data

Sending data via HTTP

Sending data is what HTTP is used for.

HTTP sets out rules for sending data.

Rules allow computers that know very little about each other to communicate without lengthy negotiations.

Sometimes HTTP is not enough. We need more rules.

Building on top of HTTP, we can send complex data — not just HTML, CSS and JavaScript.

Commonly these days we send images, videos, binary files, and even instances of objects.

Why send objects over HTTP?

Databases store a lot of the world's data.

Data are usually stored using some sort of structure, so that they're reusable.

Object usually means blueprint, the plans of the house.

Instance usually means realisation of an object, like an actual house built from the plans.

JavaScript doesn't make the distinction too clear.

Often we would like to send a small chunk of structured data from a database server to a user's machine. Then their browser cna display it nicely.

HTTP with JSON can do this.

JSON

JavaScript A scripting/programming language.

Object Groups of name–value pairs.

Notation Set of rules for representing objects.

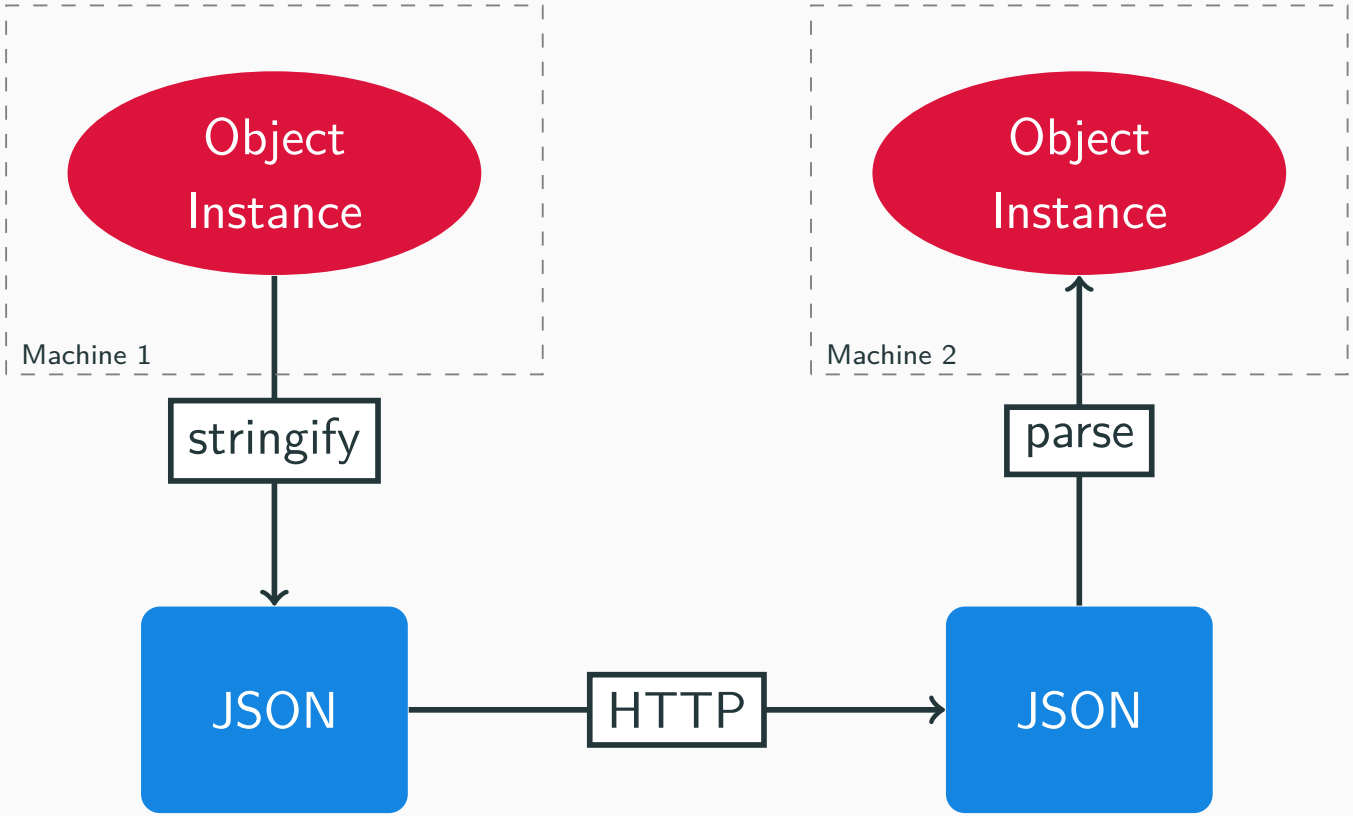
JSON is just text — text that conforms to a syntax.

Inluenced by JavaScript's syntax, but it is useable in all languages.

Represents information in text form.

Popular because it is easy to send over HTTP and parse in JavaScript.

Sending JSON



JSON Example

```
{
  "employees": [
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
  ]
}
```

Using JSON in JavaScript

The key here is that an object instance on one machine can be turned into JSON text, transmitted, and then reconstructed into an object instance on another machine.

```
// Turning text into a JavaScript object.
var obj = JSON.parse(text);
// obj is an obect instance.

// Turning a JavaScript object into text.
var text = JSON.stringify(obj);
// text is a string.
```

JSON Syntax

- Objects identified by curly braces.

```
{}
```

- Name:Value pairs within objects separated by a comma.

```
{"name": "Ian", "fingers": 10}
```

- Lists identified by square brackets.

```
["Ian", "Marco", "Sam"]
```

- All names and strings use double quotes.

```
"Ian"
```

JSON Types

- Numbers

  ```
  123.456
  ```

- Strings

  ```
  "Hello, world!"
  ```

- Boolean

  ```
  true
  ```

- Arrays

  ```
  [1,2,3]
  ```

- Objects

  ```
  {"name": "Ian"}
  ```

- null

  ```
  null
  ```

eXtensible Markup Language

XML is a JSON alternative.

eXtensible Designed to accommodate change.

Markup Annotates text.

Language Set of rules for communication.

JSON seems to be used more frequently than XML in most modern web applications.

XML seems to be a little more verbose.

XML was designed with more uses in mind.

HTML and XML look similar.

XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<book isbn-13="978-0131774292" isbn-10="0131774298">
  <title>Expert C Programming: Deep C Secrets</title>
  <publisher>Prentice Hall</publisher>
  <author>Peter van der Linden</author>
</book>
```

Using XML

Tags are not pre-defined tag names – you make them up yourself.

Syntax follows a tree-like pattern. Tags can be nested within other tags.

Document Object Model (DOM) is a concept related to XML.

XML Syntax

Declaration XML documents should have a single line at the start stating that it's XML, the version of XML it is, and an encoding.

Elements XML is structured as elements, which are enclosed in angle brackets.

Root element XML must have a single root element that wraps all others.

Attbirutes Elements can have attributes, which are name–value pairs within the angle brackets. A given attribute name can only be specified once per element.

Entity references Certain characters must be escaped with entity references, e.g. < for $\langle$.

Case sensitive Everything in XML is case sensitive.

XML Syntax Example

```
<?xml version="1.0" encoding="UTF-8"?>
<parent-element attribute-name="attribute-value">
  <child name="value">Text</child-element>
  <child name="value">Text</child-element>
  <child name="value">Text</child-element>
  <lone-warrior />
</parent-element>
```

Document Object Model

The DOM is a programming interface for HTML and XML documents.

Models the document as a structured group of nodes that have properties and methods.

Connects web pages to scripts or programming languages.

Use document.createElement, document.createTextNode and document.element.appendChild to add to the DOM.

Use document.getElementById to access elements of the DOM.

Extensively in web applications.

Angular, jQuey and React have different ideas about the developers relationship with the DOM.

developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

Using JSON and XML

JSON and XML don't do anything by themselves — they're just text.

HTTP can be used to retreive JSON (or XML) from a server.

Developers can use JavaScript to retreive JSON (or XML) in the background of a web application.

The DOM of the web page currently displayed in a browser can then be manipulated to display this information.

Web applications are really just web pages that use this trick.

The mechanism by which a web application does this is called AJAX.

Facebook uses this to display new posts once the user has scrolled to the bottom of the screen.

Asynchronous JavaScript and XML

AJAX stands for Asynchronous JavaScript and XML.

Asynchronous In the background, and without a page refresh.

JavaScript Programming language for the web.

XML eXtensible Markup Language.

Page refreshes happen when the webpage is removed from the view, and a new one is rendered.

JavaScript can be used to change this behaviour. On clicking a link, and AJAX call can be triggered instead, and the web page remains.

Location bar can even be manipulated.

More about AJAX

AJAX allows us to make a HTTP request from JavaScript, receive the response from that request and deal with it.

Despite the name, we don't have to receive XML — we can use JSON or anything else.

Calls happen asynchronously — other code can run while waiting.

HTTP requests are usually relatively slow.

Callback functions are used when the HTTP transaction is complete.

JavaScript libraries provide easy-to-use AJAX functions.

Raw AJAX Example

```
var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {
  if (xmlhttp.readyState == 4) {
    var mydiv = document.getElementById("mydivid");
    mydiv.innerHTML = xmlhttp.responseText;
  }
};

xmlhttp.open("GET", "https://goo.gl/2GCplC");
xmlhttp.send();
```

AJAX Example Explained

1. XMLHttpRequest is a built-in class that provides AJAX functionality in JavaScript.
2. httpRequest.onreadystatechange should be set to a function to run every time something happens in our HTTP call.
3. httpRequest.open is called to initialize the request.
4. httpRequest.send is used to send the request to the server.
5. XMLHttpRequest.readyState changes when the state of the AJAX call changes. This triggers a call to httpRequest.onreadystatechange.

Using jQuery

jQuery is much easier to use.

```
<script src="jquery.min.js"></script>

$.get("https://goo.gl/2GCplC", function(data) {
  $("#mydivid").html(data);
});
```

REST

REST

- REST stands for Representational State Transfer.
- REST is an architecture describing how we might use HTTP.
- RESTful APIs make use of more HTTP methods than just GET and POST.
- Most HTTP APIs are not RESTful.
- RESTful APIs adhere to a few loosely defined constraints.
- Two of those constraints are that the API is stateless and cacheable.

drdobbs.com/web-development/restful-web-services-a-tutorial/240169069

Typical example

Suppose we have a system for storing and retrieving emails.

Method	URL	Description
GET	/emails	list all emails
POST	/email	store new email
GET	/email/32	retrieve email with id 32
PUT	/email/32	update email with id 32
DELETE	/email/32	delete email with id 32

bitworking.org/news/201/restify-daytrader

Stateless

- Statelessness is a REST constraint.
- HTTP uses the client-server model.
- The server should treat each request as a single, independent transaction.
- No client state should be stored on the server.
- Each request must contain all of the information to perform the request.

Cacheable

- REST APIs should provide responses that are cacheable.
- Intermediaries between the client and server should be able to cache responses.
- This should be transparent to the client.
- Cacheability increases response time.
- Browsers usually cache resources, in case they are requested again.
- There is usually a time limit on cached resources.

Sessions

Sessions

HTTP treats each request–response individually.

How can we let users identify themselves to the server as the same user who made a previous request?

Needed to enable such things as shopping carts.

Sessions provide a mechanism for this.

Servers can provide a unique key in a response, which can be used in the next request.

Amazon were one of the first companies to design this functionality.

Security

- HTTP is not encrypted.
- HTTPS is a protocol based on HTTP, but it provides security.
- GET and POST are by far the most commonly used HTTP methods (by web developers).
- Data sent by GET and POST will be encrypted over HTTPS.
- However, it's generally accepted that POST is more secure for sending sensitive data.
- This is because browsers will typically cache and servers will typically log URLS, with the data encoded in them.

HTTP APIs

- Facebook, Google, Reddit and others often provide programmable interfaces to their services.
- This lets other application developers use the services programmatically.
- For instance, Reddit allows developers to create mobile apps for viewing and making submissions to reddit.
- HTTP is often the mechanism used for this purpose.
- Access is provided through a set of URLs, across a variety of HTTP methods.
- The APIs often require JSON in HTTP request bodies and often return the query results as JSON.

NoSQL

- NoSQL is the umbrella term for databases that do not conform to the relational, SQL-style model.
- Relational databases are good for some types of data.
- However, they have some issues.
- SQL queries can result in costly joins.
- Tables can be sparsely populated.
- Two common NoSQL database types are Document-oriented and Graph.

CouchDB

- CouchDB is a document-oriented database.
- Documents are represented in CouchDB as JSON objects.
- Each document has its own id and revision, indicated by properties `_id` and `_rev` in the JSON document.
- Updating a document leaves its `_id` intact, but updates its `_rev`.
- Different documents can have different properties – there is no schema.
- The main interface with CouchDB, for storage and retrieval is a HTTP API.
- CouchDB uses HTTP methods such as `GET`, `POST`, `PUT` and `DELETE` to retrieve, add, update and delete documents.

Futon

- CouchDB has an in-built admin interface.
- It's called Futon.
- You access it through the `/_utils` path.
- You can create and delete databases.
- You can also create, update and delete documents.

MapReduce

- MapReduce is a way of programming.
- It is a model for performing specific types of problems that are common in programming.
- MapReduce promotes algorithms that have an initially embarrassingly parallel part, and a subsequent consolidation part.
- The former is the Map part, and the latter is the Reduce part.
- MapReduce isn't necessarily anything new, the ideas have existed for a long time.
- The formalisation of those ideas and their implementation in systems such as Hadoop is useful.

joelonsoftware.com/items/2006/08/01.html

Map

Map takes a function and a list, and applies the function to every element of the list.

```
function map(fn, a) {
  r = [];
  for (i = 0; i < a.length; i++)
          r[i] = fn(a[i]);
        return r;
}
```

joelonsoftware.com/items/2006/08/01.html

Reduce

Reduce takes the output of Map, and accumulates the elements in some way.

```
function reduce(fn, a, init) {
  var s = init;
  for (i = 0; i < a.length; i++)
      s = fn(s, a[i]);
  return s;
}
```

joelonsoftware.com/items/2006/08/01.html

Map Reduce in CouchDB

Reduce takes the output of Map, and accumulates the elements in some way.

```
function(doc) {
  if(doc.date && doc.title) {
    emit(doc.date, doc.title);
  }
}

function(keys, values, rereduce) {
  if (rereduce)
    return sum(values);
        else
    return values.length;
}
```

http://guide.couchdb.org/draft/views.html

Redis