Data Representation and Querying

ian.mcloughlin@gmit.ie

Topics

About this module

HTTP

REST

JSON

XML

AJAX

CouchDB

Map Reduce

About this module

Learning outcomes

On completion of this module the learner will/should be able to:

- Explain the benefits and the limitations of a variety of data models.
- Determine the most appropriate data model given a set of requirements.
- Represent, integrate and querying large datasets using existing API's and frameworks.
- Describe the principles behind both the linked data and the open data movements.

Examinations

Type	%	Date
Project	50	Week 8
End of Semester Exam	50	See exams timetable

HTTP

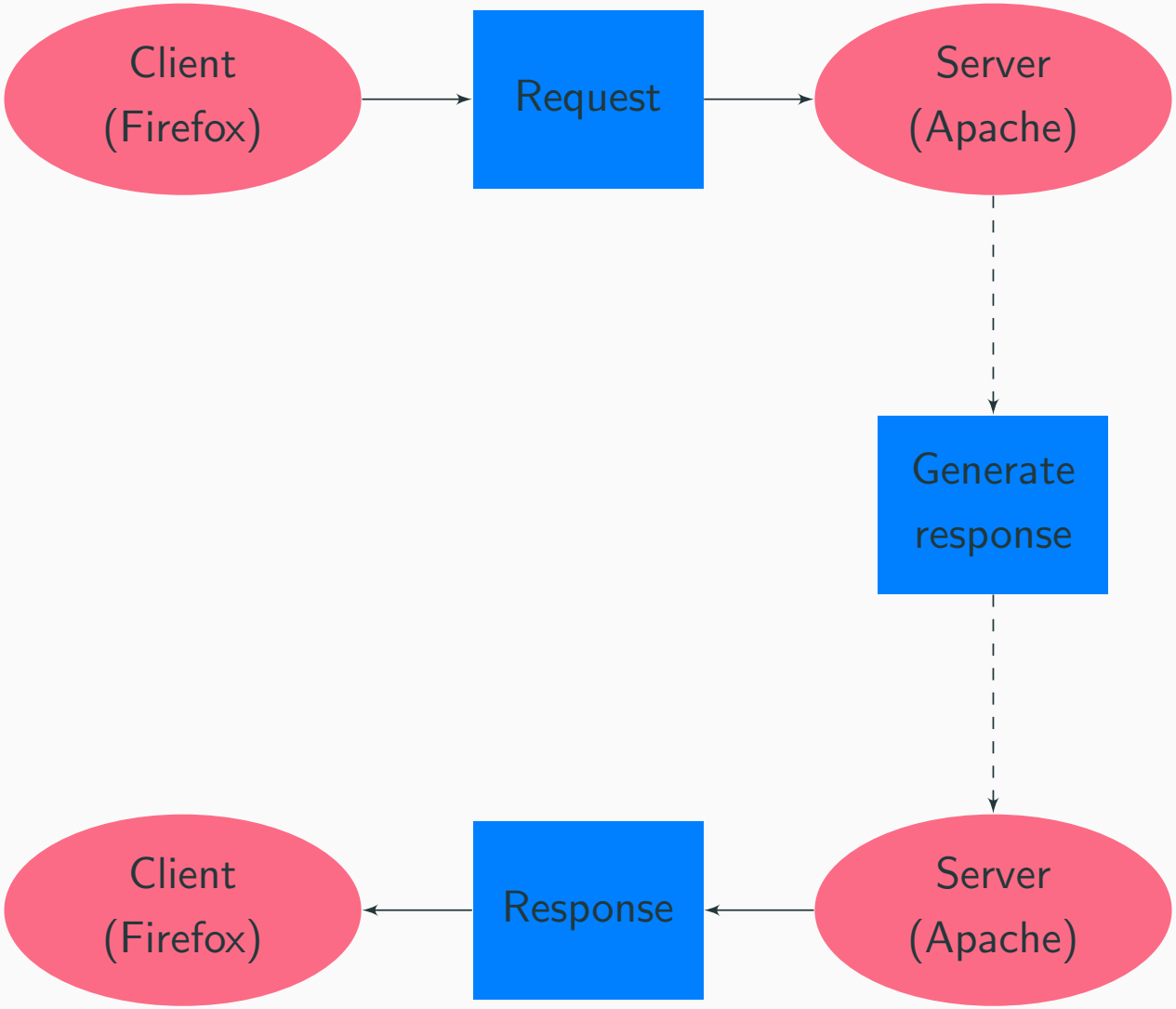
HyperText Transfer Protocol

HyperText Text with links.

Transfer Communication of data.

Protocol Set of rules for communication.

Request–Response

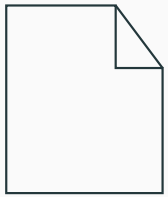


Uniform Resource Locator

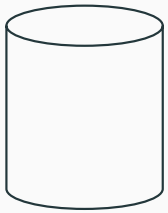
http://www.reddit.com:80/r/funny/?limit=1

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

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.

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 server will typically log URLS, with the data encoded in them.

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 /path/file.html HTTP/1.0
From: someuser@jmarshall.com
User-Agent: HTTPTool/1.0
```

Request (POST)

```
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
```

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>
```

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 Parameters go in the URL after ?, e.g. http://www.google.ie?q=Funny+cats.

POST Parameters go in the body.

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 cachable.

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

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.

Cachable

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

Example

Suppose we have a system for storing and retrieving member's details.

Method	URL	Description
GET	/people	list members
POST	/people	create member
GET	/people/1	retrieve member
PUT	/people/1	update member
DELETE	/people/1	delete member

bitworking.org/news/201/restify-daytrader

JSON

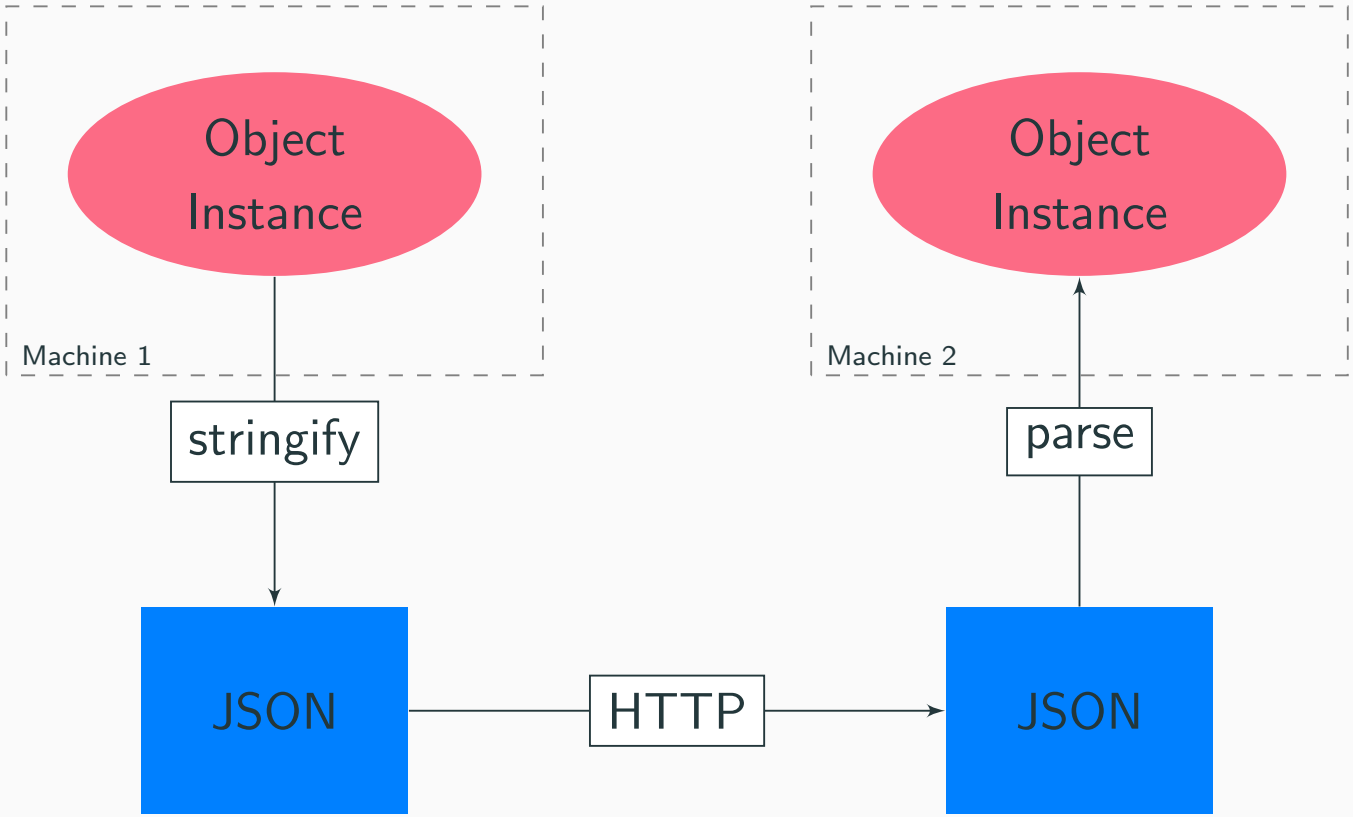
JSON

JavaScript A scripting/programming language.

Object Groups of name–value pairs.

Notation Set of rules for representing objects.

Sending JSON



JSON Example

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

Using JSON in JavaScript

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

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

JSON Syntax

- Name/Value pairs separated by a colon.

  ```
  "name": "Ian"
  ```

- Objects identified by curly braces.

  ```
  {}
  ```

- Lists identified by square brackets.

  ```
  []
  ```

- All strings (and names) use double quotes (not single).

  ```
  "Ian"
  ```

JSON Types

- Numbers

  ```
  123.456
  ```

- Strings

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

- Boolean

  ```
  true"
  ```

- Arrays

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

- Objects

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

- null

  ```
  null
  ```

XML

AJAX

NoSQL

Map Reduce