

Data Representation and Querying

ian.mcloughlin@gmit.ie

About this module

HTTP

REST

JSON

XML

HTTP APIs

AJAX

NoSQL

Map Reduce

About this module

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.

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

HTTP

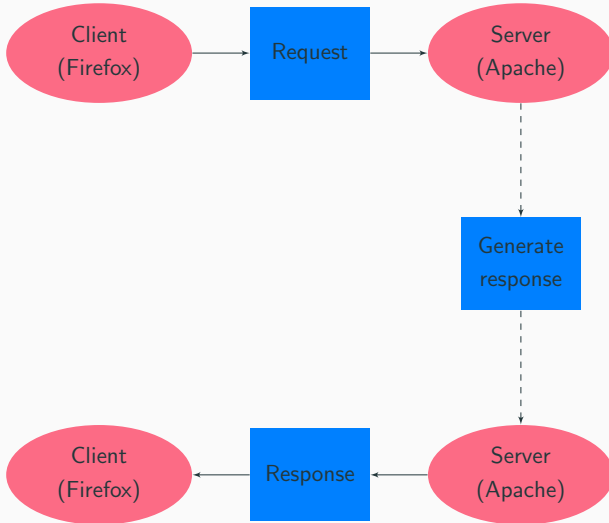
HyperText Transfer Protocol

HyperText Text with links.

Transfer Communication of data.

Protocol Set of rules for communication.

Request-Response



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

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



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 . . .

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.

- 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.

Requests and responses both have this format:

- Initial 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>

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 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.

- 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.

- 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.
- 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

JSON

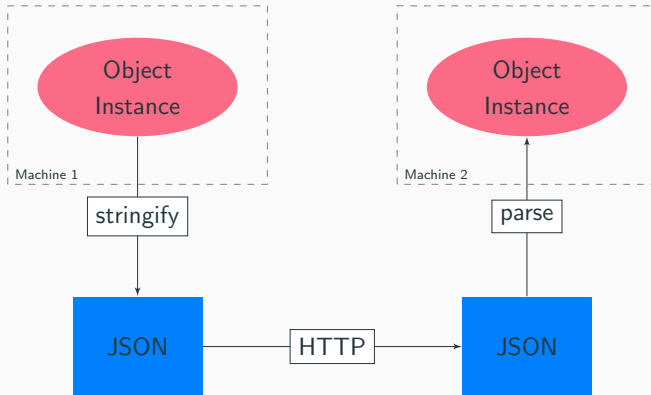
JavaScript A scripting/programming language.

Object Groups of name–value pairs.

Notation Set of rules for representing objects.

- JSON is just text, but text that conforms to a syntax.
- JSON is heavily influenced by JavaScript, but it is used in with all languages.
- JSON's primary purpose is to represent information in text form.
- JSON is 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

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

- 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

Extensible Designed to accommodate change.

Markup Annotates text.

Language Set of rules for communication.

- XML is an alternative to JSON.
- XML looks like HTML, but it is different.
- XML's purpose is to represent information in text form.
- There are no pre-defined tag names – you make them up yourself.
- XML has a tree-like syntax.
- The Document Object Model (DOM) can be applied to XML.

```
<?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>
```

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.

Attributes 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 <.

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>
```

HTTP APIs

AJAX

NoSQL

Map Reduce