SWEN 301 : Scalable Software Development

Web Application Development - Introduction

Jens Dietrich **(jens.dietrich@vuw.ac.nz)**

Purpose and Summary

This is part 1 of an architecture case study, covering history, use cases and design goals for web applications.

The technical focus of the case study is on web (server) application development using the Java Enterprise Edition -- starting with traditional web applications to dynamically generate HTML pages, leading to the development of HTTP services and how this underpins the development of modern, distributed serviced-based applications

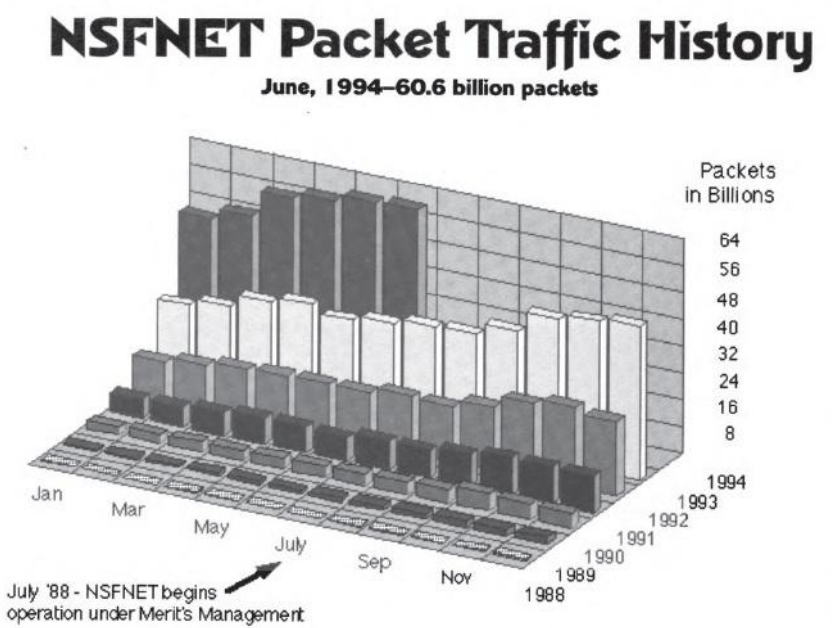
John Gage (SUN Micro) 1984 :

The network is the computer

The Early History of the Web

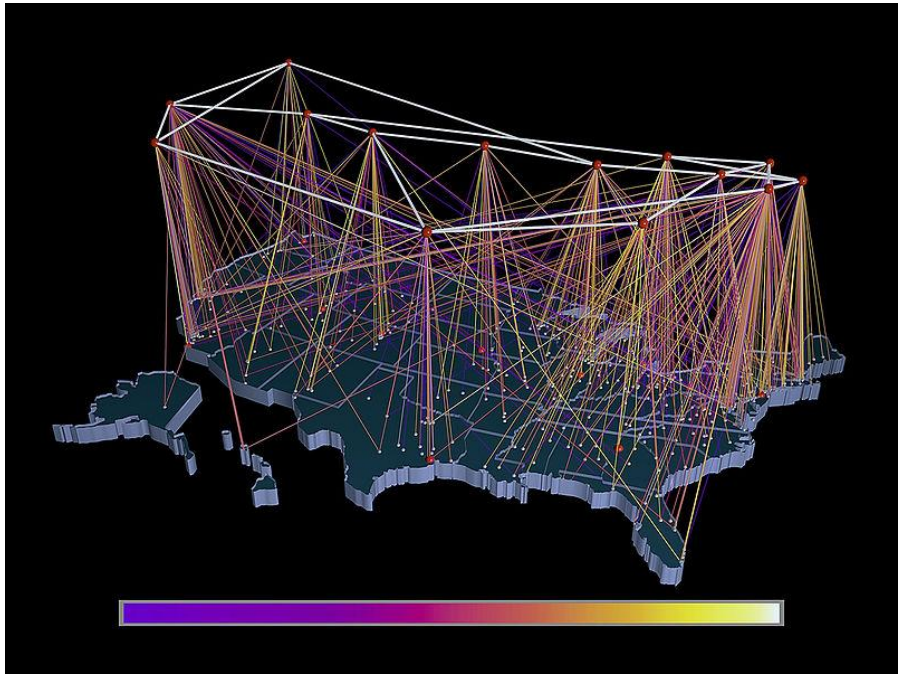
- advanced Research Projects Agency Network (ARPA), was a wide-area network starting around 1968
- used routers
- initially it had four nodes (UCLA, UCSB, Stanford, Univ. of Utah)
- expanded to the US east coast around 1970
- TCP/IP protocol used from around 1975
- replaced by NSF National Science Foundation Network (NSFNET) starting around 1985

The Early History of the Web



http://en.wikipedia.org/wiki/File:NSFNETTrafficGraph-June1994.jpg

The Early History of the Web



NSFNET Traffic 1991, NSFNET backbone nodes are shown at the top, regional networks below, traffic volume is depicted from purple (zero bytes) to white (100 billion bytes), visualization by NCSA using traffic data provided by the Merit Network.

http://en.wikipedia.org/wiki/File:NSFNET-traffic-visualization-1991.jpg

The Rise of HTTP/HTML

- CERN (European Organization for Nuclear Research) started using TCP/IP around 1984
- information sharing
- Tim Berners-Lee proposed the World-Wide Web in 1989 based on the idea of hypertext
- hypertext was seen as more suitable to share information than directories (trees) and by keyword

Tim Berners-Lee, CERN: Information Management: A Proposal. March 1989, http://www.w3.org/History/1989/proposal.html

Static Web Pages: whitehouse.gov, 1994

http://www.telegraph.co.uk/technology/internet/10663451/The-early-days-of-25-websites.html
http://archive.org/web/ (the way back machine)

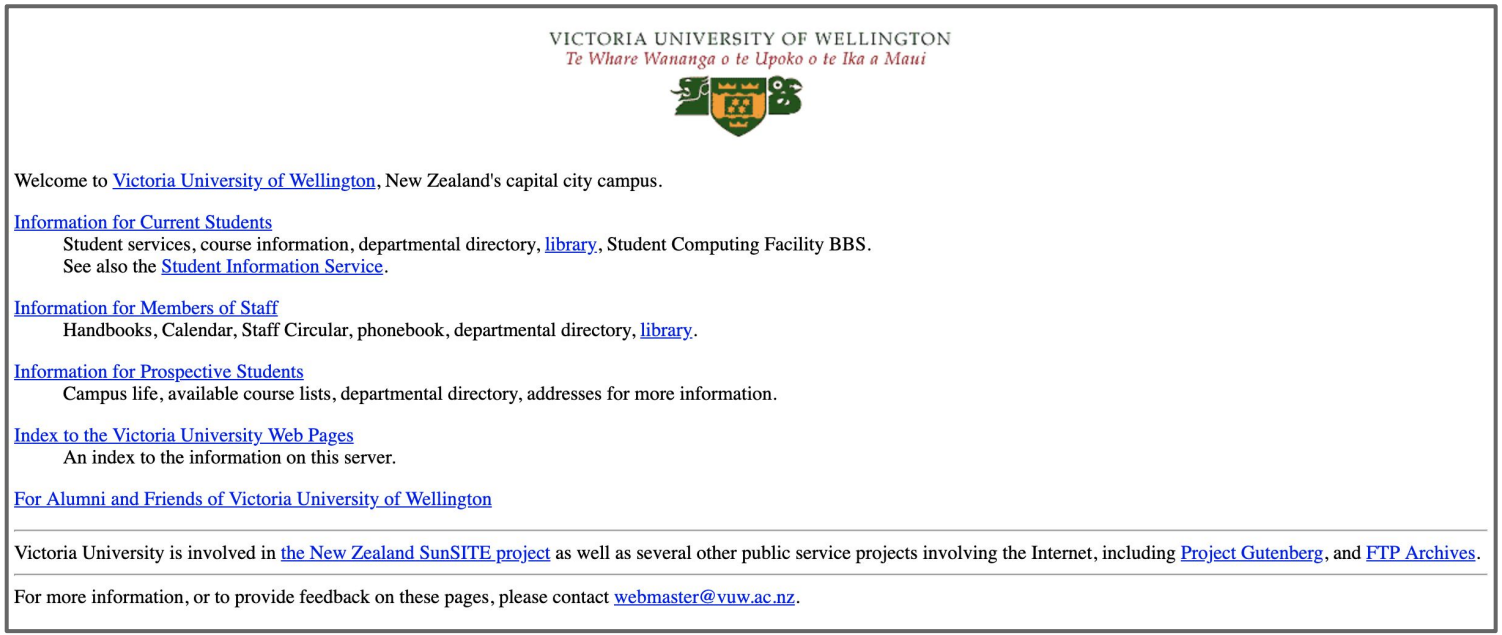
The WWW Standards

three standards were developed at CERN to facilitate the exchange of hypertext document based on TCP/IP networks:

- **HTTP** - application protocol
- **HTML** - document format, including support for links between documents
- **URL** - uniform resource locators - a global naming scheme used to identify documents and other resources

Vic's Website, 27 April 97

https://web.archive.org/web/19970515000000*/http://vuw.ac.nz/



VICTORIA UNIVERSITY OF WELLINGTON

Te Whare Wananga o te Upoko o te Ika a Maui

Welcome to Victoria University of Wellington, New Zealand's capital city campus.

Information for Current Students
Student services, course information, departmental directory, library, Student Computing Facility BBS.
See also the Student Information Service.

Information for Members of Staff
Handbooks, Calendar, Staff Circular, phonebook, departmental directory, library.

Information for Prospective Students
Campus life, available course lists, departmental directory, addresses for more information.

Index to the Victoria University Web Pages
An index to the information on this server.

For Alumni and Friends of Victoria University of Wellington

Victoria University is involved in the New Zealand SunSITE project as well as several other public service projects involving the Internet, including Project Gutenberg, and FTP Archives.

For more information, or to provide feedback on these pages, please contact webmaster@vuw.ac.nz.

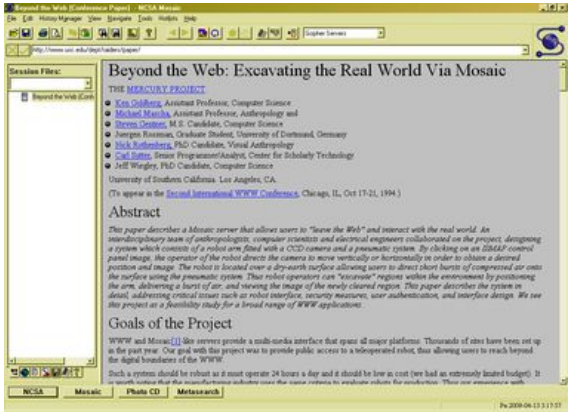
Setting Standards

to ensure the openness of the internet, standards are maintained by several government- and industry-independent standard organisations, such as:

- the **Internet Engineering Task Force** (**IETF**), since 1986, focuses on low-level standards (in particular TCP/IP)
- the **World Wide Web Consortium (W3C)**, since 1994, is responsible for many “high-level” WWW standards including HTTP, URL/URI, HTML, XML and CSS.
- the **European Computer Manufacturers Association**(**ECMA**) - standardised JavaScript and several other programming languages and file formats

Early Tools (Client): Mosaic

- Mosaic was the first widely-used graphical web browser
- developed at the National Center for Supercomputing Applications (NCSA) at the University of Illinois Urbana-Champaign 1992-97
- supported WWW protocols, and older protocols like gopher and FTP
- free for non-commercial use



http://en.wikipedia.org/wiki/File:NCSA_Mosaic.PNG

Early Tools (Client): Netscape Navigator

- first mainstream browser, 1994-98
- first commercial success story, company values at 3 billion USD in 1995, creating the first internet millionaires
- rapidly lost market share in the late 90ties in the browser wars to MS Internet Explorer
- open sourcing started in 1998, resulted in Mozilla/Firefox

http://en.wikipedia.org/wiki/File:OS2_Netscape_Communicator_4.61.png

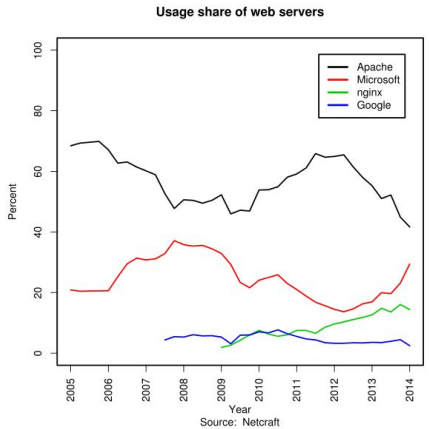
Early Tools (Client): Netscape Navigator (ctd)

Netscape invented several features which are crucial for many web-based business models:

- JavaScript - the ability to make web pages dynamic
- applets - applications can be embedded in web sites (collaboration with SUN)
- HTTPS - encrypted connections to transmit sensitive information like credit card data
- cookies

Early Tools (Server): NCSA HTTPd

- early generation web server developed in 1993-96 by NCSA
- when development stalled in 1995, this formed the base for the free and open source Apache web server (with a reference to “another patchy web server”)
- the Apache open source foundation founded in 1999



http://en.wikipedia.org/wiki/File:Usage_share_of_web_servers_(Source_Netcraft).svg

Business Models Emerging

- first commercial ISPs (1989)
- doubleclick - advertising (1995)
- amazon - book store (1995)
- hotmail - web-based email (1996)
- ebay - online auction site (1996)

Requirements: Linking the Web to the Back Office

- to businesses like Amazon or eBay, static web pages were not sufficient
- web pages had to be generated from back office data, often stored in relational databases (RDBMS)
- RDBMS support transactions and concurrency control - a requirement for many business processes

The Rise of CGI

- common gateway interface (CGI) are scripts to dynamically create web pages
- scripting languages like PERL were used for early CGI scripts
- this technology was developed in the mid 90ties, and supported by NCSA httpd and the apache server
- a Java-based CGI technology is **Servlets (1997)**

Example: PERL CGI Script

```
#!/usr/bin/perl
 print "Content-type: text/html\n\n";
 print <<HTML;
 <html>
 <head>
 <title>A Simple Perl CGI</title>
 </head>
 <body>
 <h1>A Simple Perl CGI</h1>
 <p>Hello World</p>
 </body>
 HTML
 exit;
```

Features Needed for a Usable Web

- the rise of web-based businesses created new requirements for web applications, such as
- **sessions** - adding state to web applications
- **encryption** - secure communication to protect sensitive data
- **compression** - compress (text-based) data to improve scalability
- **caching** - use locally stored data whenever possible

Transparency

- starting in the early 90ties, various technologies were developed to support these services
- in many cases, this is **transparent** for the programmer, and can be achieved through configuration (of the web server)

Sessions

- http is a **stateless** protocol: pairs of isolated http requests and responses
- applications like eBay and Amazon needed to support **sessions**
- multiple user requests are interconnected, and result in one business transaction
- example: amazon shopping carts

Sessions (ctd)

- browser support: cookies (starting with Netscape 0.9, 1994)
- when revisiting web pages, a unique cookie is sent to the server
- the server can use this cookie to associate several request/response pairs into one unit, and associate resources (like the shopping card) with them
- cookies are secure (size and number is restricted, and cookie data is not interpreted), but have caused privacy concerns
- cookie-less session tracking is possible via URL rewriting (appending session ids to URLs)

Compression

- web pages are char based documents
- excellent compression ratios can be achieved with standard algorithms (gzip etc)
- http headers support a **content-encoding** header (RFC2616, 1999)
- if the server and the browser both support this, compression is applied automatically and *transparent* to both
- this significantly increased the **scalability** of the web

Encryption

- secure communication is needed to protect sensitive data
- Netscape started implementing **Transport Layer Security** (**TLS**) and its predecessor, **Secure Sockets Layer** (**SSL**) around 1994
- http using these techniques is called https (s for secure)
- these technologies *transparently* encrypt communication between browsers and web servers

Encryption (ctd)

- browsers can authenticate servers based on sent certificates, and can establish whether they trust the server based on whether the certificate was issued by a trusted certification authority (CA)
- this is based on the assumption that the certification authority has ("manually") checked the identity and trustworthiness of (company running the) server
- the first commercially successful CAs were VeriSign and Thawte

Caching

- caching creates local or near-local copies of documents and resources (such as images) to accelerate subsequent access to the same resource
- early browsers like Netscape Navigator already supported caching
- proxy servers can cache resources for several clients within an organisation
- the first proxy servers appeared in the mid 90ties (the open source squid, from 1996)

Caching (ctd)

- content delivery networks (CDN) implement caching on a larger scale, by copying information to a set of distributed servers
- clients can then access resources from nearby servers
- one of the first large-scale CDNs was Akamai, starting around 1998
- *transparency* is crucial for caches, otherwise the wrong (version of) data is retrieved

Content Creation: Logic vs Content

- in the 90ties, creating interactive web pages was a programming intensive tasks
- many web sites created by programmers were not attractive, reflecting the fact the programmers are not designers
- what was needed was technologies that **separated logic and content from presentation**, so that different people with different skill sets could work in parallel on these aspects

CSS

- stylesheet technology emerged between 1994 and 96 as Cascading Style Sheets (CSS)
- MSIE adopted CSS in 96, and the W3C started standardising it in the same year
- CSS enables designers to define the visual web appearance of elements in a web site, without requiring much knowledge about the programmatic logic

From CGI to Server Pages

- document-centric technology to create web pages, better fit for designers
- i.e., instead writing a script to create a web page, write a web page (in HTML) with embedded scripts to compute parts of the document dynamically (at request time)
- this principle is called a server page , it is based on templating
- example: PHP (around 1995), Active Server Pages (ASP, 1997), **Java Server Pages (JSP)**, 1999

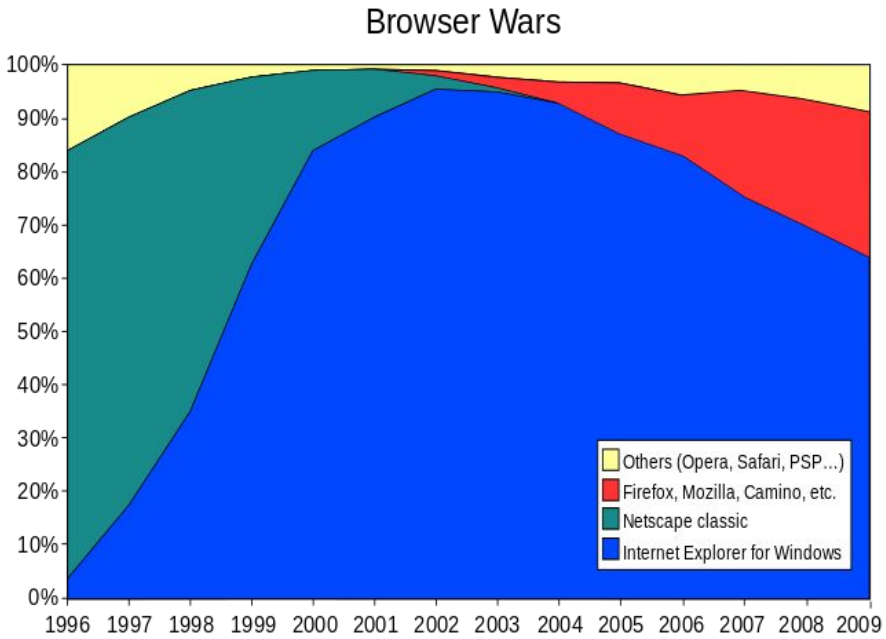
Towards Interactive WebPages

- the first-generation web pages were slow and unresponsive
- to change the information displayed in the browser, users had to send a HTTP request to the server, and wait for the response to be displayed
- to address this, Netscape developed JavaScript, a light-weight programming language running in the web browser
- JavaScript can be used to manipulate web pages

The Browser Wars

- the early years of the internet were characterised by open standards and freely available tools
- this changes when Microsoft launched MSIE in 1995
- bundled with Windows, it started to dominate the market
- led to technology fragmentation and stagnating standards

Market Share of Web Browsers, 1996-2009 and 2019



http://en.wikipedia.org/wiki/File:Browser_Wars_(en).svg

2009 - current:

https://gs.statcounter.com/browser-market-share#monthly-200901-202203

Browser Wars and Web Development

- the browser wars made web development more expensive and hindered innovation
- developers had to write code (client and server) that distinguished between separate browsers with different levels of standard compliance
- MSIE become the nightmare of web development
- this was one reason “cross-platform” libraries (like JQuery, from 2006)

Post Browser Wars

- Netscape survived the browser war as open source project (Mozilla/ Firefox)
- MSIE started to decline around 2005
- in 2010, Firefox again reached around 30% market share
- WebKit is a rendering machine that can be seen as a browser core that originated from the KHTML (part of the KDE Linux desktop)
- WebKit is now used by two browsers now dominating (due to their use in mobile OSs): Safari and Chrome
- diverse browser market, better standardisation and compatibility, WebKit-based browsers dominate

Java

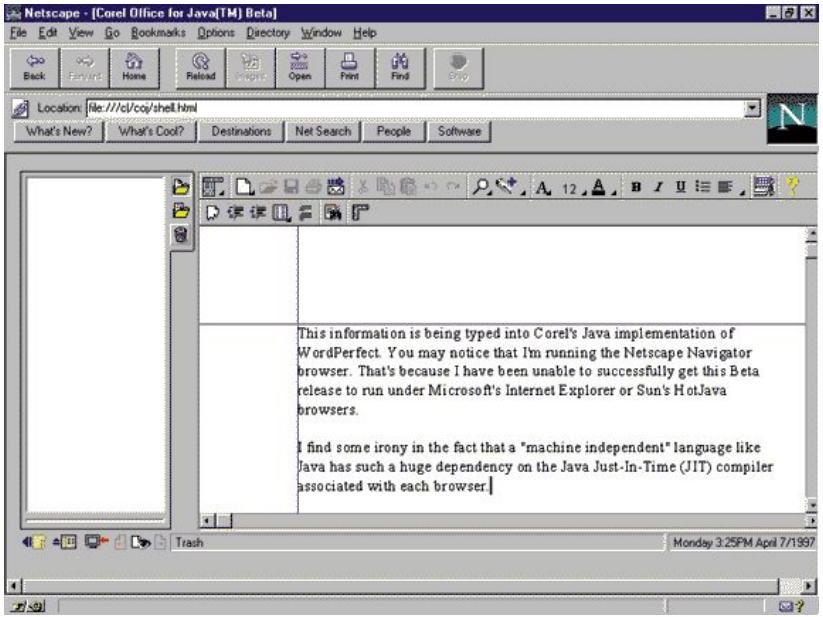
- Java was developed by SUN Microsystems in the early 90ties, JDK 1.0 was released in 1995
- in the beginning, Java was aimed at programming TV set top boxes
- Java code was platform independent
- Netscape Navigator 2.0 (1995) had support for executing Java programs embedded in web sites (applets)

Applets

- applets are Java programs embedded in HTML sites
- the `<applet>` element was used
- use cases included simple animations, games, and full-fledged applications
- applets could be executed on different platforms (Windows, Solaris and various other Unix derivatives, Mac)
- a strict security model (sandbox) restricted applets
- e.g., applets can not access the local file system

Office in The Browser ? !

Corel Office for Java was an applet-based office suite published in 1997



http://www.drdobbs.com/corel-office-for-java/184415588

Why Applets Failed

- initially, MSIE supported Java
- MS boycotted applets, tried to fork Java
- gave rise to alternative technologies: Macromedia/Adobe Flash scripting was based on ECMA Script (called ActionScript): started in 96, peaked around 2010-15
- Similar: MS Silverlight, JavaFX
- HTML5 (from 2014) added similar capabilities to HTML/JS

Ajax

- Microsoft included a **`XMLHttpRequest`** object in MSIE in 1999, and other browsers quickly added support
- with **`XMLHttpRequest`**, it is possible to send asynchronous requests to the server and partially update the document (=DOM), instead of replacing the entire document
- the W3C standardised **`XMLHttpRequest`** in 2006
- initially, XML was often used to encode data, this has since shifted to JSON

Ajax (ctd)

- this technology became known as **AJAX** (Asynchronous JavaScript and XML)
- AJAX was made popular by its use in Gmail (2004) and Google Maps (2005)
- this type of application (where the entire page never changes but is only updated) is often called **single-page web application (SPA)**
- a consequence was that web applications were written to produce structured data, instead of entire web pages

Push vs Pull

- the architecture of the web is pull-focused
- HTTP: clients pull information from the server
- there are many use cases for a push based model, e.g., displaying the latest news without the need to refresh the page
- there are various Ajax-based solution all based on the idea of continuously querying the server (aka “long poll”)
- HTML5 has two technologies to directly address this feature: server-sent events and websockets

Web Services

- writing web server applications that do more than just dishing out web pages
- clients other than the browser can use these **web services** as well
- use cases:
  - use services in 1-page web applications in browsers (e.g., with Ajax)
  - use services in mobile apps
  - use services in general applications, including server applications
- leads to modern service-based architectures (SOA)
- services provide a high level of **transparency**:
  - location -- logical location separated from physical location via DNS
  - implementation -- use whatever language
  - platform -- host on whatever platform
  - standardised security

Web Services (ctd)

- early attempts (around 2000): SOAP, WSDL, UUDI: complex, XML-based, not successful
- REST-based -- based on original HTTP methods, usually encoding data in JSON, now widely used
- REST was proposed by Roy Fielding (part Kiwi) in 2000
- now organisation drive development from web apis: specification of REST endpoints developers can use to integrate services (e.g. shopping with Amazon, github, slack, export to google spreadsheets, ..)
- from 2018: attempts to standardise API development: OpenAPI / Swagger