UNIVERSITY OF TECHNOLOGY SYDNEY

WEB SYSTEMS

Tom McBride
University of Technology, Sydney

UTS: ENGINEERING AND INFORMATION TECHNOLOGY

The internet

- The invention of the internet has been a significant technological advance
- Web technologies have caused many fundamental changes in many industries
- Web-based systems are ubiquitous and share some common features, some common challenges

In this lecture

- A brief history of the internet
- Web application qualities
- Implementing web applications
- Communication between components
- Sample web applications

UNIVERSITY OF TECHNOLOGY SYDNEY

What is the web and how did it all start

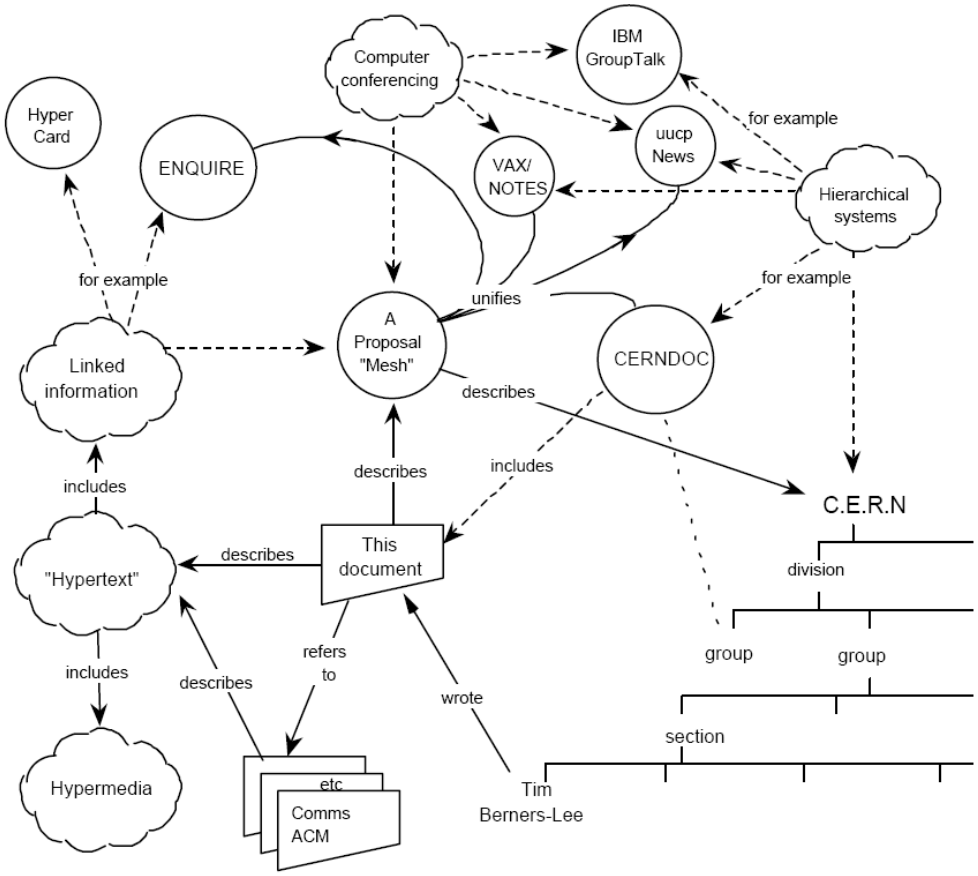
A BRIEF HISTORY

A bit of history

- ARPANET was created to provide the a packet switched network (~1969)
- Network robustness driven by unreliable nodes (not nuclear attack)
- TCP/IP implemented 1983
- Tim Berners-Lee implemented a network based hypertext in 1989
- Mosaic web browser introduced 1993, Netscape in 1994
- Then things exploded, kind of

Berners-Lee original proposal

- Apple had Hypercard, an application enabled a “stack” of cards to be linked in various ways.
- Berners-Lee had written a multi-user version “Enquire”
- Now wanted to create a universal linked information system



HTTP is stateless

Therefore: use cookies as a session key

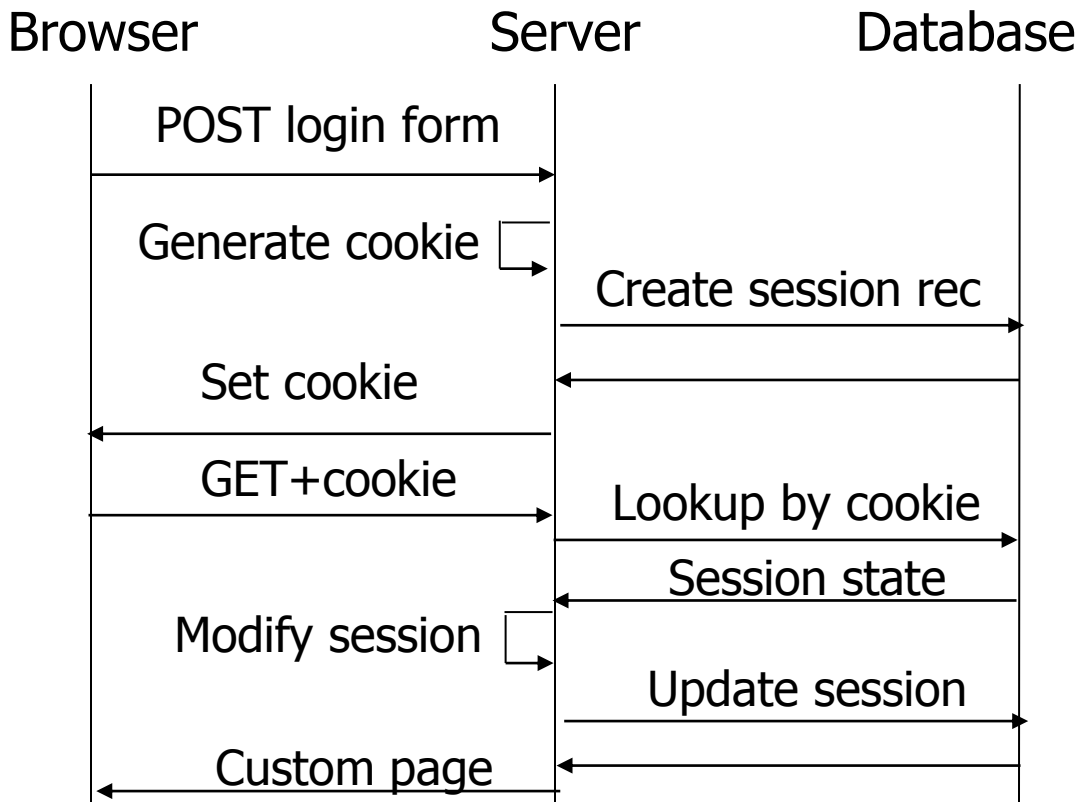
Browser

XYZ_ID=atyugo

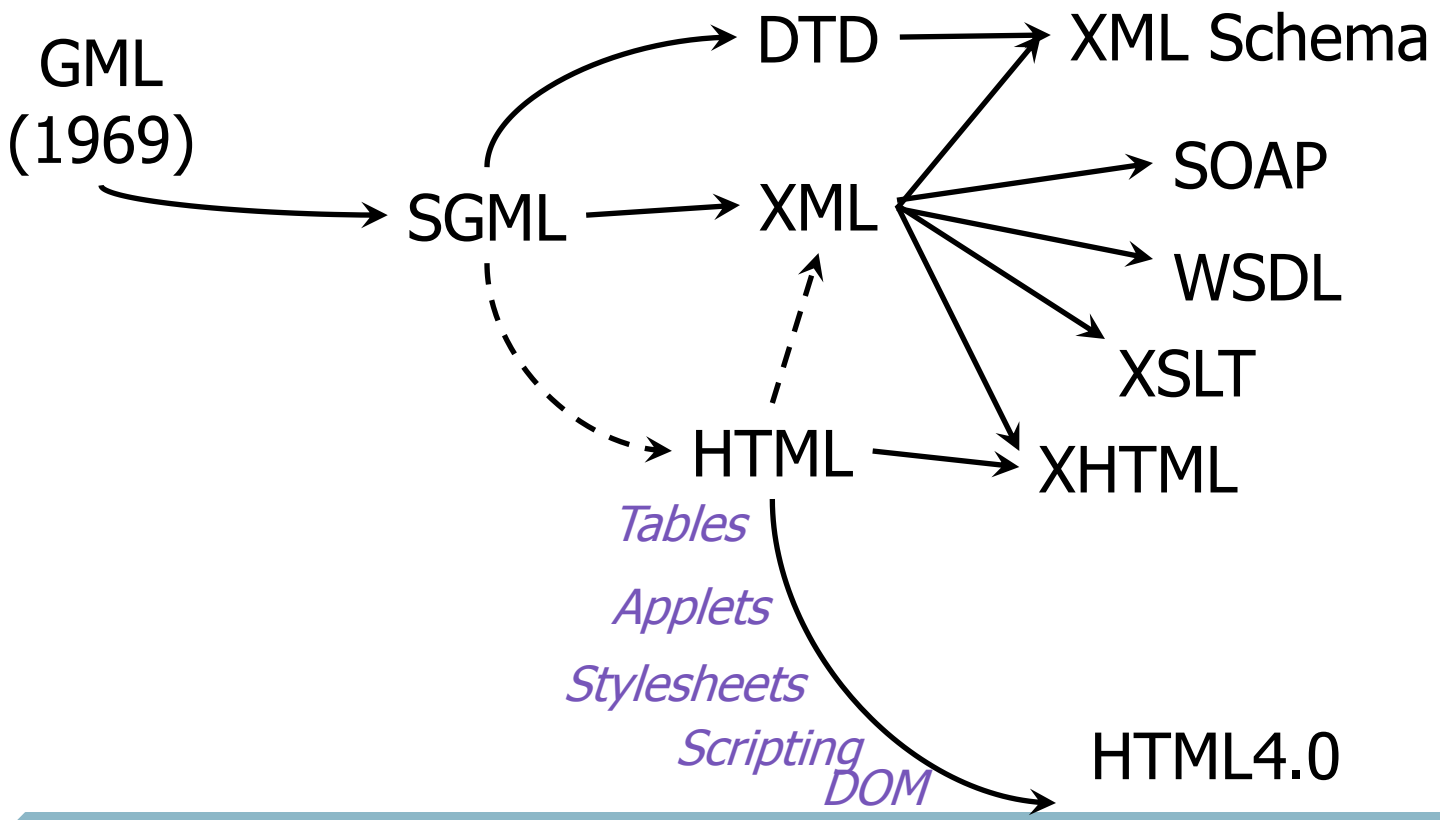
Server

Database

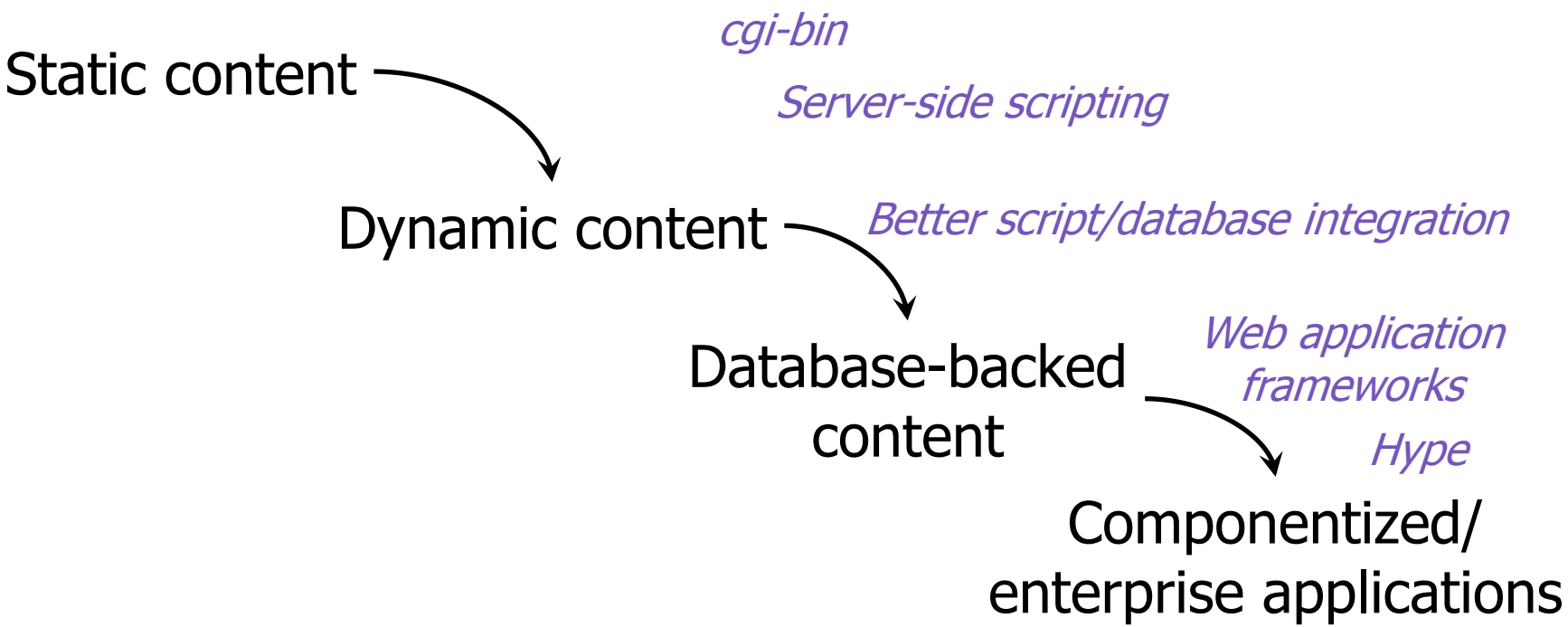
sesskey	*userid*
atyugo	johnr



The origins of Hypertext Markup Language



From linked static information to …



Definitions

- “"Web Services" is the umbrella term of group of loosely related Web-based resources and components that may be used by other Web applications over HTTP. Those resources could include anything from phone directory data to weather data to sports results. “
  - http://www.sitepoint.com/glossary.php?q=W
- “Web services let computers talk to one another over the Internet, allowing computer programs to exchange information by eliminating barriers such as different hardware platforms, software languages, and operating systems that usually make different programs incompatible. Web services make it easier to share information, data, and services, as well as making it cheaper and easier for businesses to work with on-line partners. This technology will help to increase e-business.”
  - http://www.albertasupernet.ca/general/glossaryp-w.htm

UNIVERSITY OF TECHNOLOGY SYDNEY

COMMUNICATIONS BETWEEN COMPONENTS

Three competing approaches

- XML-RPC – Remote Procedure Call using XML
  - Requests use HTTP POST
  - Simple XML payload for request and response
- REST – Representational State Transfer
  - Requests use HTTP GET and POST
  - Parameters encoded as HTTP query parameters
- SOAP – Simple Object Access Protocol
  - Requests can use “any” transport, but HTTP POST is often used as an example
  - Not-so-simple XML payload for request and response

The basic concept

Request Message body contains an XML request packet instead of parameters	**POST /my/resource/ HTTP/1.0** **Host: myserver.com** **<?xml version =“1.0”?>** ***<xmlrequest>*** ***<xmltag><morestuff>…</morestuff></xmltag>*** ***</xmlrequest>***
Response Message body contains an XML response packet instead of HTML	HTTP/1.0 200 OK Content-Type: text/xml Content-length: 746 <?xml version =“1.0”?> *<xmlresponse>* *<xmltag><morestuff>…</morestuff></xmltag>* *</xmlresponse>*

Why this is interesting…

- Arbitrarily complex requests and responses can be layered over HTTP
  - XML is a computer-readable data format
  - This is just as much distributed computing as CORBA/RMI/DCOM
- Existing HTTP servers can handle this traffic
  - Provision of vast amounts of data in “computer-readable” form (XML) as well as “human-readable” form (HTML)
  - Integration of open services with existing (Web) infrastructure

XML-RPC

- "Remote Procedure Call"
- Examples from www.livejournal.com
- A URI is simply an entry point to a set of methods provided by the service
- Request body conforms to a simple XML format
- Parameters are passed in order

```
POST /api/RPC2 HTTP/1.0
Host: plant.blogger.com
Content-Type: text/xml
Content-length: 515

<?xml version="1.0"?>
<methodCall>
  <methodName>blogger.newPost</methodName>
  <params>

<param><value><string>744145</string></value>
</param>

<param><value><string>ewilliams</string></value
></param>
  </params>
</methodCall>
```

XML-RPC (2)

- Response body also conforms to a simple standardized XML format

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 125
Content-Type: text/xml

<?xml version="1.0"?>
<methodResponse>
  <params>

<param><value><string>4515151</string><
/value></param>
  </params>
</methodResponse>
```

Examples from www.livejournal.com

REST

- “Representational State Transfer”
- Coined by Roy Fielding in about 2000
- Refers to the fact that HTTP is fundamentally stateless --- every transaction carries its state token with it
- Focus on **Resources** and **Interface**

GET /my/resource/?user=JohnR HTTP/1.0
Host: myserver.com

HTTP/1.0 200 OK
Content-Type: text/xml
Content-length: 746

```
<?xml version =“1.0”?>
<userdata>
 <username>JohnR</username>
 <address>Factory of Engineering</address>
</userdata>
```

REST (2)

- Services are accessed as resources (URIs)
- GET reads data only
- POST modifies data only
- REST has no standardized XML format, although specific forms are appearing (RSS, Atom)
- REST requests can be examined by the firewall
- Uses Web servers for authentication and authorization

POST /my/resource HTTP/1.0
Host: myserver.com
user=JohnR&address=Factory%20of%20IT

HTTP/1.0 200 OK
Content-Type: text/xml
Content-length: 254

```
<?xml version =“1.0”?>
<response status=“updateok”>
  <username>JohnR</username>
</response>
```

SOAP

- A complex set of specifications (still evolving)
- HTTP is only a “transport”
- Emphasis on typing, service definition
- Commonly generated by higher-level tools (eg .NET Web Services)

- Examples from www.flickr.com

```
POST /objectURI HTTP/1.1
Host: www.flickr.com
SOAPMethodName: urn:flickr#flickr.echo
Content-Type: text/xml
Content-Length: 777

<s:Envelope
    xmlns:s=http://www.w3.org/2003/05/soap-envelope
    xmlns:xsi=http://www.w3.org/1999/XMLSchema-
instance
    xmlns:xsd="http://www.w3.org/1999/XMLSchema" >
  <s:Body>
    <x:FlickrRequest xmlns:x="urn:flickr">
      <method>flickr.echo</method>
      <name>value</name>
    </x:FlickrRequest>
  </s:Body>
</s:Envelope>
```

SOAP (2)

- WSDL describes services (like an IDL for SOAP)
- UDDI supports registration and discovery of services
- Examples from www.flickr.com

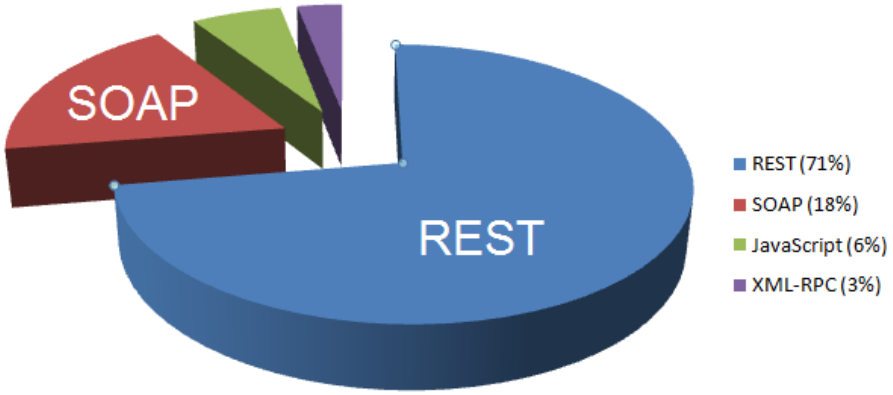
```
HTTP/1.1 200 OK
Connection: close
Content-Length: 944
Content-Type: text/xml

<s:Envelope>
  <s:Body>
    <x:FlickrResponse>
      <method>flickr.test.echo</method>
      <format>soap</format>
      <foo>bar</foo>

<api_key>36f59f1f07d65be29a85196cf3260c1c</api_k
ey>
    </x:FlickrResponse>
  </s:Body>
</s:Envelope>
```

XML-RPC vs REST vs SOAP

	XML-RPC	REST	SOAP
Services	Request and receive information	Get, Post, Put, Delete	Application defined
Advantages	Simple, allows complex data structures to be transmitted, processed and returned	Lightweight – not a lot of extra xml markup. Human readable results. Easy to build – no toolkits required	Easy to consume – sometimes. Rigid – type checking, adheres to a contract. Development tools
Standards		Data format is not standard	WSDL, XML, SOAP envelopes
API flexibility	Simple	Simple, accessible URI based	Requires specific knowledge of XML
Bandwidth usage	Light	Light	Verbose
Security		Administrator or firewall can discern message intent	Application responsible for authentication & authorization

Summary

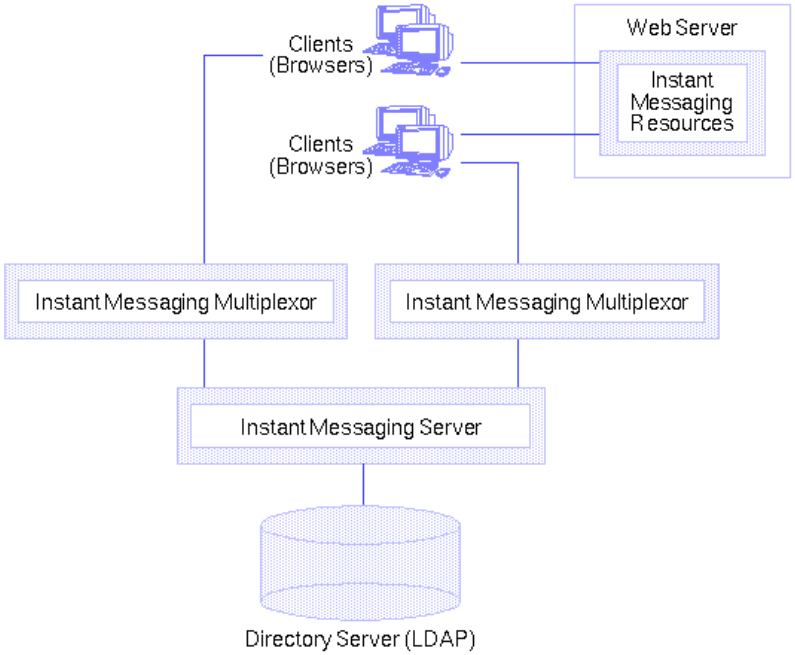
- There are three communication protocols between web service components; XML-RPC, REST, SOAP
- XML-RPC suits small, simple applications.
- REST and SOAP seem equally capable but not equally favoured



IMPLEMENTING WEB APPLICATIONS

Implementing web applications

- An HTTP handler on the front end
- Almost any programming language behind that…
- And “a” database
- Loosely coupled components, connected using specific protocols

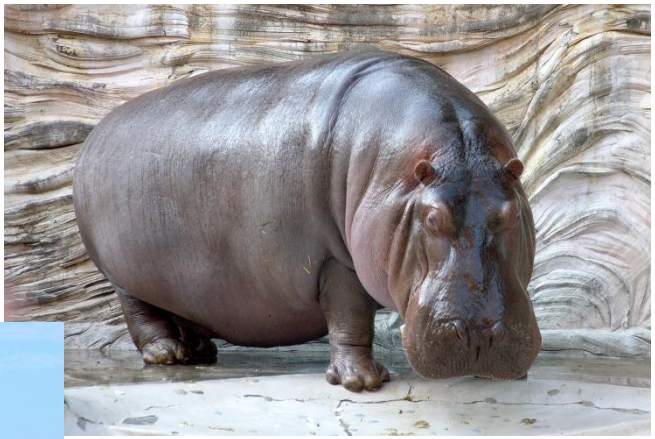


Coupling and cohesion

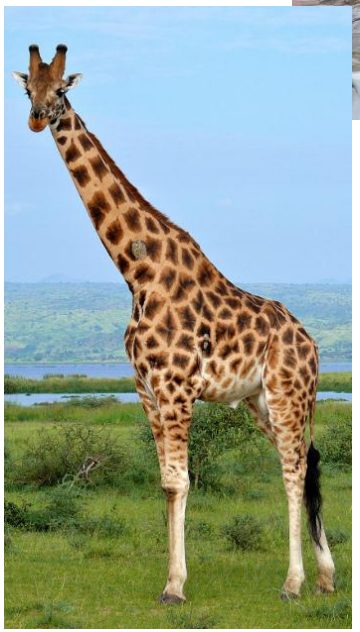
- A web system is just like any other system except that connections between modules can be message-based and comparatively slow
- After that comes considerations of scaling, security and performance
- Pay attention to coupling and cohesion principles

Fat client- thin client

- Fat client puts most of the processing with the end user
  - Little central data
  - Lot of processing (well, almost)
  - Example – Loan calculations

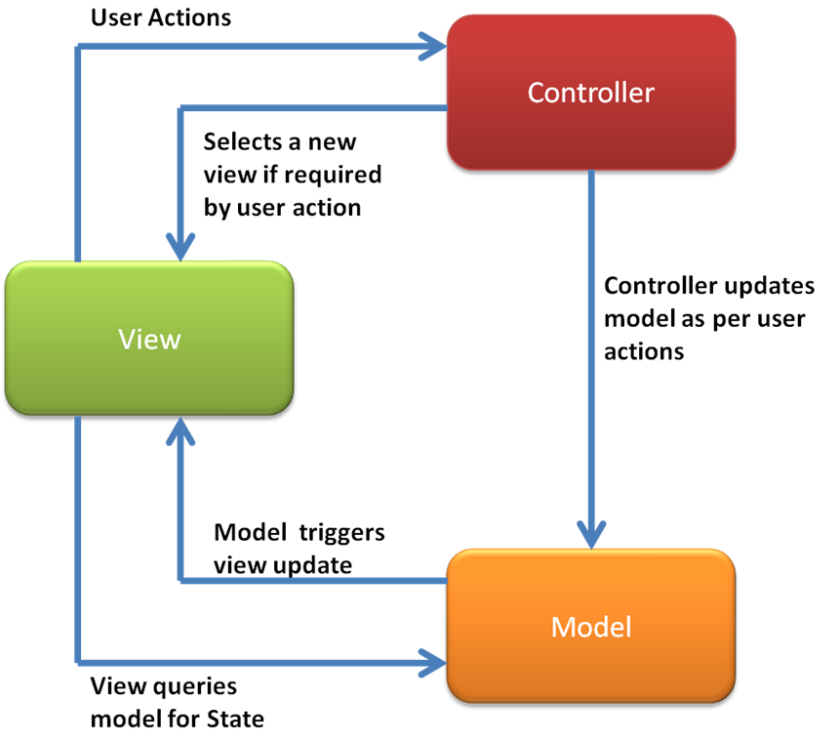


- Thin client puts most of the processing at the server
  - Little user input
  - Lots of centralised data
  - lots of processing
  - Example – reservation system, classified ad placement



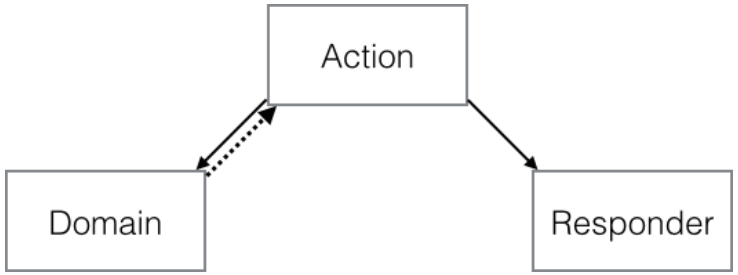
MVC and variants

- Model – View – Controller is the most common web application architecture
- Allows the same information to be presented in different ways
- Popular applications;
  - Ruby On Rails: A popular Ruby web-framework
  - Apple Cocoa: Apple's framework for developing Mac OS and iOS applications
  - ASP.Net Framework: Microsoft's web-framework for implementing web applications
  - Apache Struts: A popular Java web-framework



Action – Domain - Responder

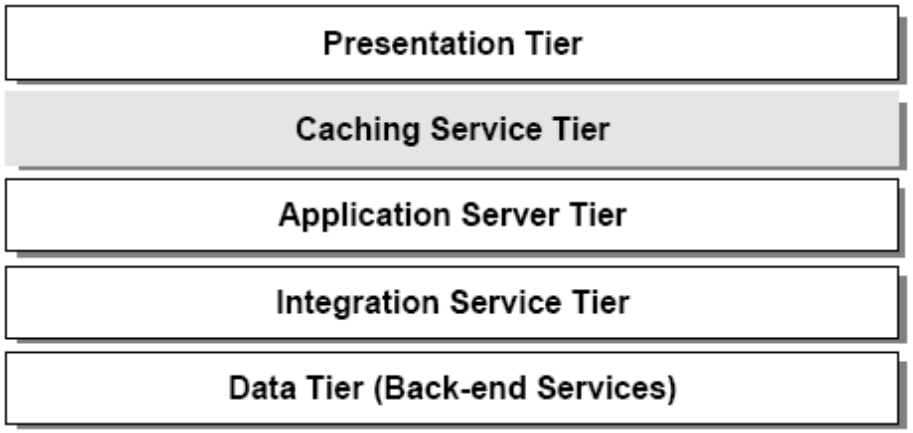
- *Action* is the logic that connects the *Domain* and *Responder*. It uses the request input to interact with the *Domain*, and passes the *Domain* output to the *Responder*.
- *Domain* is the logic to manipulate the domain, session, application, and environment data, modifying state and persistence as needed.
- *Responder* is the logic to build an HTTP response or response description. It deals with body content, templates and views, headers and cookies, status codes, and so on.
- Think of this as the user seeing multiple objects, each able to invoke some action and get a response independently of other objects.



Scalability can be achieved with n-tier architectures

- N-tiered architecture
  - Allows different components to be run on different servers
  - A tier is a functionally separated hardware and software component that perform as a specific function
  - Highly flexible because each tier can be scaled independently
  - Communication between the tiers through standard protocols

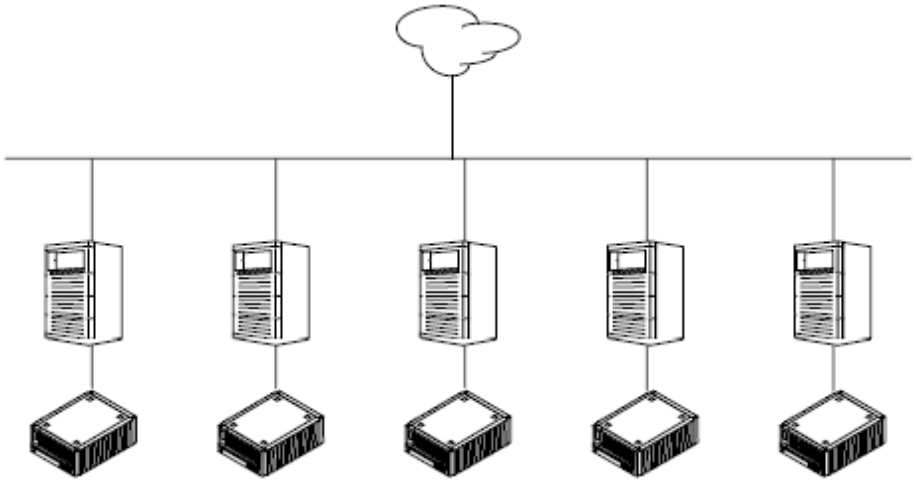
A common N-tier architecture layer approach



Horizontal scaling

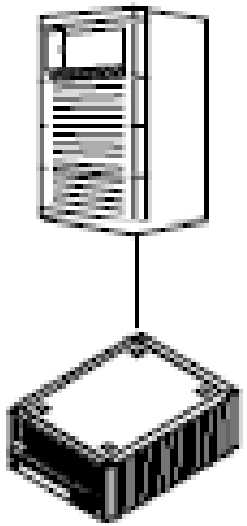
- Robust – if one server fails it can be taken offline without disrupting the overall service
- Use a load balancer to improve resource utilization
- Works best when there is minimal state information and scaling is easy

Horizontal scaling employs multiple servers within a tier



Vertical scaling

- With vertical scaling, resources are scaled within the system.
- Characterised by slow changing applications
- Often necessary for data intensive services
- Items that cannot be partitioned should be close together to ensure latency and uniform access



Don’t re-invent the scalable web architecture

- Most hardware vendors have web server products
  - Sun – Sun Cluster 3.0
  - IBM – ranger of server architectures
  - Microsoft – software solutions
  - Sure to be more
- J2EE vs .NET
  - Web services frameworks and tools

Which qualities matter and how are they achieved?

WEB APPLICATION QUALITIES

Quality attributes of web-based systems

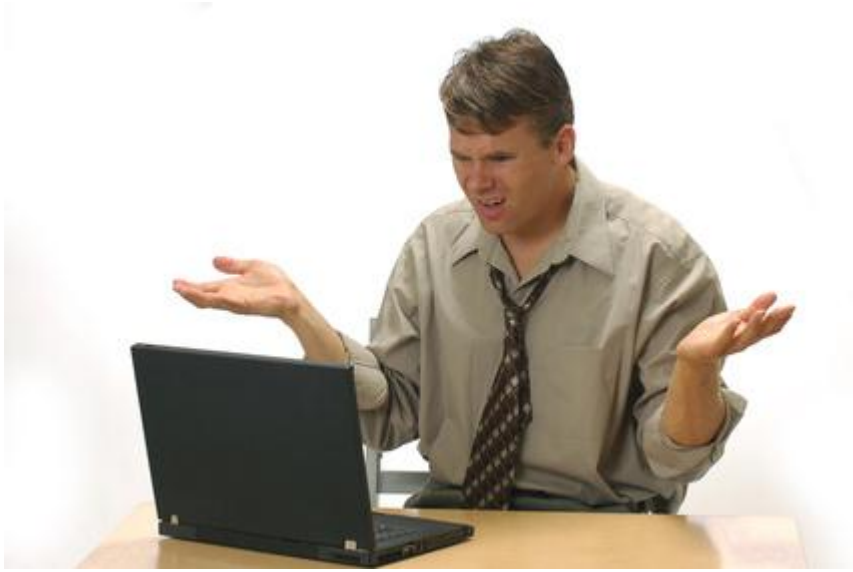
- Web applications need to be
  - Scalable
    - Growth is limited by global use, not organizational growth
    - Very hard to test load capacity
  - Secure
    - Even public applications like Facebook have security issues
    - Data theft emerging as a major issue
  - Usable
    - Converging to some common interface themes and metaphors
  - Reliable
    - Truly 24*7 in some cases

Security

- Not all users are well intended
- Need to balance access with safety
- Applications have become the easier target to attack than servers
- Limit what the application can do
- Take lessons from other domains
  - Concert halls and stadia limit where the audience can go and limit their movements. Everyone gets a seat, no-one can move too far, too fast.
  - Separate potential antagonists
  - Carry an ID card (security token) and check it

Usability on the web

- Neilsen's heuristics have become popular
  - Visibility of system status
  - Match between system and the real world
  - User control and freedom
  - Consistency and standards
  - Error prevention
  - Recognition rather than recall
  - Flexibility and efficient use
  - Aesthetic and minimalist design
  - Help users recognise, diagnose and recover from errors
  - Help and documentation



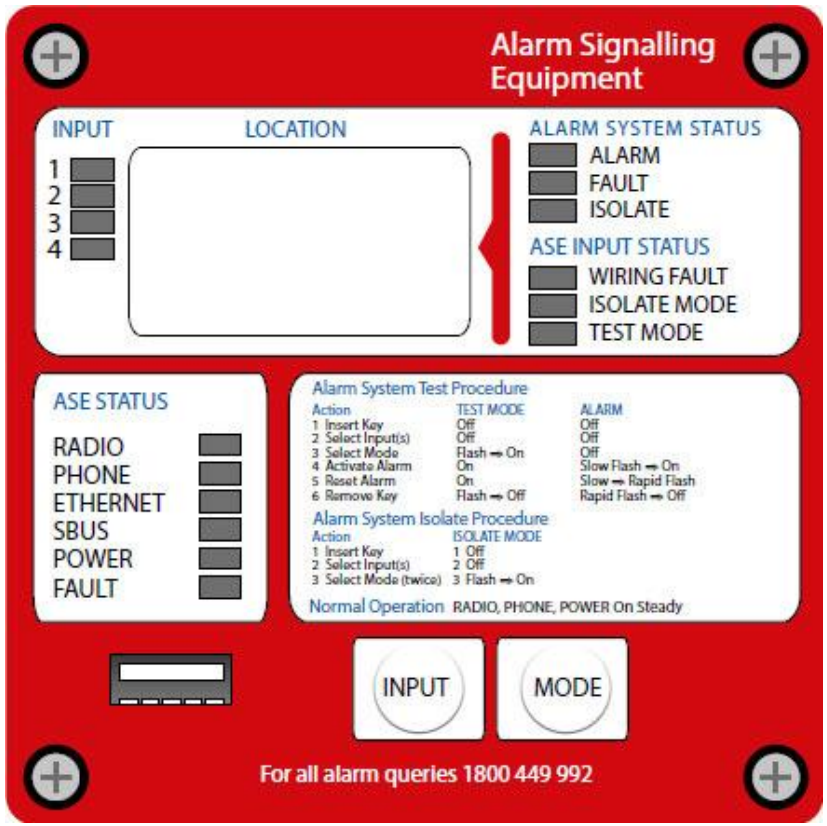
Usability examples

- Lots of advice available
- Users hate learning
  - Allow users to enter data in a familiar way. E.g. 612 9154 4528, + 61 2 9514 4528
- Users hate repeating
  - If data has been entered once, retain it.
  - Default values should reflect majority choice
- Users hate waiting
  - Show progress bars
  - Show partial results

Source: http://www.sitepoint.com/article/architecture-usable/

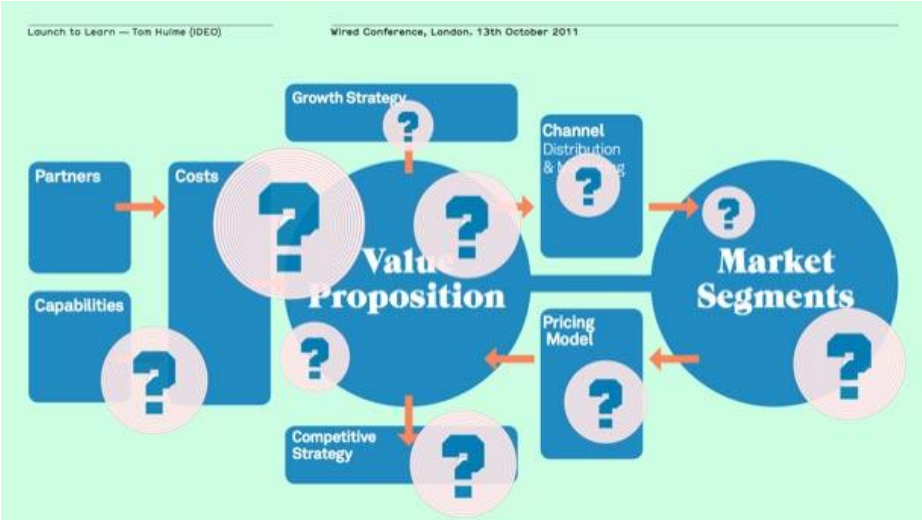
Reliability

- Web systems tend to be dynamic, evolving
- Higher likelihood of defects than life critical systems
- Can’t always guarantee transaction completion
- Allow a user to check and recover
- High reliability architecture needs multiple things to fail before the whole system fails (multiple points of failure)



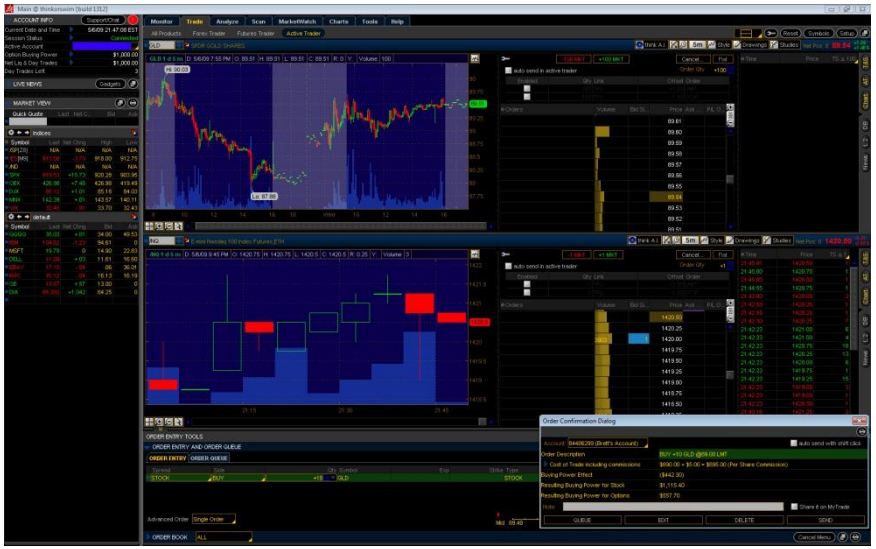
Modifiability

- Web systems evolve and change
- Highly changeable because business models are changing
- Modular architecture supports modification
- Modular systems are also more scalable



Performance

- Throughput vs. response
- Transaction throughput can be important
  - Major bank vs. local bookstore
  - Event management or reporting systems may need to cope with a web storm
  - Scalability tends to be solved through hardware but requires separable processes
- Response time needs to satisfy the user but is not required to be microsecond
- If the transaction is complex, keep the user informed



UNIVERSITY OF TECHNOLOGY SYDNEY

The rapidly expanding world of the web

A SAMPLE OF WEB APPLICATIONS

Web 2.0

- The term **Web 2.0** is commonly associated with web applications that facilitate interactive information sharing, interoperability, user-centered design,[1] and collaboration on the World Wide Web.
- A Web 2.0 site gives its users the free choice to interact or collaborate with each other in a social media dialogue as creators (prosumer) of user-generated content in a virtual community, in contrast to websites where users (consumer) are limited to the passive viewing of content that was created for them.
- Examples of Web 2.0 include
  - social-networking sites,
  - blogs, wikis, video-sharing sites,
  - hosted services, web applications,
  - mashups and folksonomies.
- a "piece of jargon" said Tim Berners-Lee "that's what was intended all along"

E-Commerce

- Buying and selling of products or services over electronic systems such as the Internet and other computer networks

- Examples
  - electronic funds transfer,
  - supply chain management,
  - Internet marketing,
  - online transaction processing,
  - electronic data interchange (EDI),
  - inventory management systems, and
  - automated data collection systems.

- Business to business (B2B), to consumer (B2C)

- Government to Government (G2G), to Business (G2B) to consumer (G2C)

Software as a service

- Software that is deployed over the internet and/or is deployed to run behind a firewall on a local area network or personal computer.

- SaaS characteristics include:
  - Network-based access to, and management of, commercially available software
  - Activities managed from central locations rather than at each customer's site, enabling customers to access applications remotely via the Web
  - Application delivery typically closer to a one-to-many model (single instance, multi-tenant architecture) than to a one-to-one model, including architecture, pricing, partnering, and management characteristics
  - Centralized feature updating, which obviates the need for end-users to download patches and upgrades.
  - Frequent integration into a larger network of communicating software—either as part of a mashup or a plugin to a platform as a service

- Examples
  - Basecamp, Huddle and other collaboration sites
  - Skype
  - Accounting services, other hosted services

The Internet of Things

- Network of physical objects or "things" embedded with electronics, software, sensors and connectivity to enable it to achieve greater value and service by exchanging data with the manufacturer, operator and/or other connected devices.
- Each thing is uniquely identifiable through its embedded computing system but is able to interoperate within the existing Internet infrastructure
- Enables Smart Cities, smart metering, remote controlled domestic devices and a lot more
- With great power comes great vulnerability

Cloud computing

- Cloud computing relies on sharing of resources to achieve coherence and economies of scale, similar to a utility (like the electricity grid) over a network.
- At the foundation of cloud computing is the broader concept of converged infrastructure and shared services.
- Using computing resources “out there” on demand.
- Cloud resources are usually not only shared by multiple users but are also dynamically reallocated per demand.
- For example, a cloud computer facility that serves European users during European business hours with a specific application (e.g., email) may reallocate the same resources to serve North American users during North America's business hours with a different application (e.g., a web server).
- Growing number of cloud-based services, e.g. Xero accounting, Trello task management, Adobe Creative Cloud, etc.

Concluding remarks

- Web computing is the most spectacularly successful set of client-server technologies ever, and will stay that way for a while.
- A wide range of application complexities is supported by current web architecture
- Emerging web protocols will likely supplant binary protocols such as CORBA and RMI for the majority of large-scale distributed applications
- Software as a service
- Cloud computing

Online resources

- "An overview of HTTP"
  http://icm.mcs.kent.edu/reports/2000/ICM-200009-0006.pdf
- "What's This All About, Then?" (HTTP tutorial)
  http://apache-server.com/Podium/show.php?p=HTTP
- HTTP 0.9
  http://www.w3.org/Protocols/HTTP/AsImplemented.html
- HTTP 1.0
  http://www.w3.org/Protocols/HTTP/HTTP2.html
- HTTP 1.1
  http://asg.web.cmu.edu/rfc/rfc2616.html
- Overview of REST
  http://en.wikipedia.org/wiki/Representational_State_Transfer
- XML-RPC
  http://www.xmlrpc.com/
- SOAP tutorials
  http://www.soaprpc.com/tutorials/