

COMP60370 Architecture

Bijan Parsia

<http://cs.man.ac.uk/~bparsia/comp60370/arch.html>

{bparsia@cs.man.ac.uk}

Stepping Back

- Thus far, focus on a document format (HTML)
 - Uncoordinated authoring
 - Validation
 - Error handling
 - Universality and longevity
 - Separation of structure and style
 - Semantic markup
 - (We've barely mentioned hypermedia!)
- Still need:
 - The Web
 - Applications

The Client

- What are the phases?
 - Retrieval
 - Parsing
 - Rendering
 - Interaction
- What are the technologies?
 - HTTP (and URI)
 - Tag soup/HTML5 to DOM
 - Could include some Javascript
 - HTML, CSS & Javascript
 - HTML, Javascript & CSS

The Server

- What are the phases?
 - Parse request
 - Gather response components
 - Massage components
 - Respond
- What are the technologies?
 - HTTP (and URIs)
 - Arbitrary
 - Arbitrary
 - HTTP

Web Differences

- Compare with:
 - Chat
 - Mail
- Some key features
 - Hyperlinks
 - Distributed documents/data units
 - Crosses organization boundaries
 - Uncoordinatedly
 - Seamless experience
 - Links take you anywhere
 - *Data* addressing
 - Not mailboxes/users

Explaining The Web

- What needs explaining?
- What sorts of explanation?
- Who needs the explanations?
- What is done with these explanations?

One Sort Of Explanation

- What needs explaining?
 - The "success" of the Web
 - Why HTTP + URI + HTML "works"
- What sorts of explanation?
 - "Web architecture" "supports"
 - Web scale
 - Web growth
- Who needs the explanations? (Purportedly)
 - Protocol and format designers
 - Application builders (and framework builders)
- [Web architecture](#) and [REST architectural style](#)

A Weak Explanation

- Metcalfe's law
 - The value of a network increases with the square of the (compatible) participants
 - Some [debate](#) on the [factor](#)
 - A corollary: large networks are expensive to abandon, thus hard to compete with
 - (But what is the [realizable](#) value?)
 - (What about [tipping points](#)?)
- "Metcalfe's Law provides much of the explanation of the success of the Web relative to earlier hypertext systems like HyperCard, Intermedia, and NoteCards. They were all much better than the Web and had features ten years ago that we are still sorely missing on the Web. But the Web was universal and the other systems were proprietary. You know who won." --[Nielson](#)

A Weaker Explanation

- [Knowledge representation (KR)] made sense but was of limited use on a small scale, but never made it to the large scale. This is exactly the state which the hypertext field was in before the Web. Each field had made certain centralist assumptions ... which prevented them from spreading globally....**The Semantic Web is what we will get if we perform the same globalization process to KR that the Web initially did to Hypertext" -- [Berners-Lee](#)**
- **The essential property of the World Wide Web is its universality.** Like the Internet, the Semantic Web will be as decentralized as possible. Such Web-like systems ... provide benefits that are hard or impossible to predict in advance. Decentralization requires compromises: **the Web had to throw away the ideal of total consistency of all of its interconnections, ushering in the infamous message "Error 404: Not Found" but allowing unchecked exponential growth. --[Berners-Lee et al](#)**

Architecture

- An abstract description of a system
 - Abstraction hides details
 - Layered architecture allows for different structures at each layer
 - Asynchronous messaging on top of synchronous one
 - Phased architecture allows for different structures at each phase
- Architecture describes
 - *elements in configurations* under *constraints* which have ("induce") *properties*
 - The properties are what we are trying to explain in terms of the architecture

Elements

- Components
 - Data processors
- Connectors
 - Data bus
- Data
 - Information transmitted across connectors and manipulated by components
 - Components and connectors can be data (and vice versa)
 - Easily seen in phased architecture
 - E.g., A program is data for an interpreter; an interpreter is a program

Configurations And Properties

- Configurations
 - Arrangement of elements, or
 - a set of constraints on component interaction
 - e.g., permitted inputs, outputs, sources, and targets
- Properties
 - Functional
 - Computing payroll, sending email, etc.
 - Non-functional
 - Scalability, extensibility, reliability

Architectural Style

- An abstract description of architectures
 - "A coordinated set of architectural constraints"
 - Elements are characterized in terms of roles or features
 - Configurations are loosely constrained
 - E.g., pipe and filter *style* constrains the "shape" of the connectors (i.e., linear, stateless, irreversible);
 - particular systems may have an architecture that conforms to this style
 - but they'll also select specific filters and connection types
- "Induction" of properties may be foiled
 - by specifics of the realization
 - dangerous fuzziness here
 - consider scalability vs. worst case complexity

Desirable (NF) Properties

- Performance
 - User perceived (esp, latency, see World Wide Wait and completion)
 - Network efficiency (esp. *cost*)
- Scalability
- Simplicity
- Modifiability
 - Evolvability, extensibility, customizability, configurability, resuability, (and I say) portability
- Visibility
 - I.e., to mediators for performance (caching) or security (firewalls)
- Reliability

Explanada

- Network-based Applications
 - Network-Based (according to Fielding)
 - Interaction between components by message passing
 - The network interactions are *perceptible*
 - Contrasted with *distributed* systems
 - Where the network is *inperceptible*
 - Application
 - Concerned with "business logic"
 - Typically heavily end-user oriented

The Web As Application

- The Web as a "Shared Information Space" (SIS)
- Requirements
 - Low-Entry Barrier
 - For authors, readers, publishers, tool/infrastructure builders
 - At the start, similar for every category
 - As things grew, barriers shifted
 - Extensibility
 - The Web changed frenetically
 - The Web *needs* to change
 - Distributed Hypermedia
 - Internet-scale
 - Esp. uncoordination
 - Positive *and* negative

Some Questions

- Fielding says: "By its nature, user actions within a distributed hypermedia system require the transfer of large amounts of data from where the data is stored to where it is used. Thus the Web architecture must be designed for large-grain data transfer." pg. 68.
 - Why the move from *amount* to *grain*?
- Is "anarchic scalability" actually a requirement?
 - Could Google run the Web?
 - What about large Walled Gardens (e.g., Facebook?)
- What about flash?

Style Derivation

- Styles are sets of constraints
 - Thus may be *composed*
 - Or, styles may be *derived* (from other styles)
 - (Presumably, not *arbitrary* merged! *Coordinated* constraints, eh?)
- REST is a derived style:
 - Client-server +
 - **Stateless** +
 - Cache +
 - **Uniform Interface** +
 - Layered System +
 - Code-on-demand (for clients)

REST Diagram

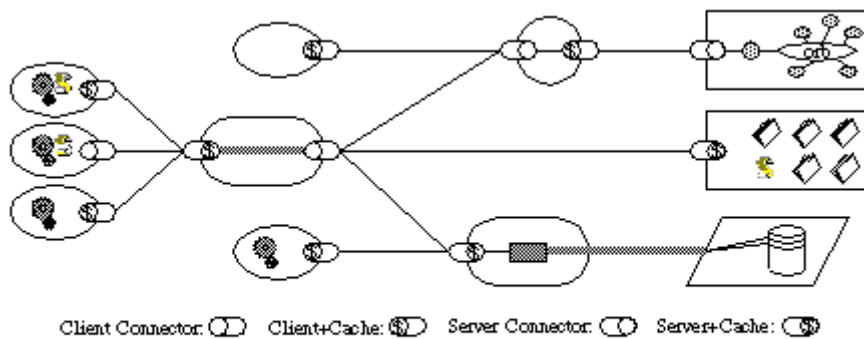


Figure 5-8. REST

REST Derivation

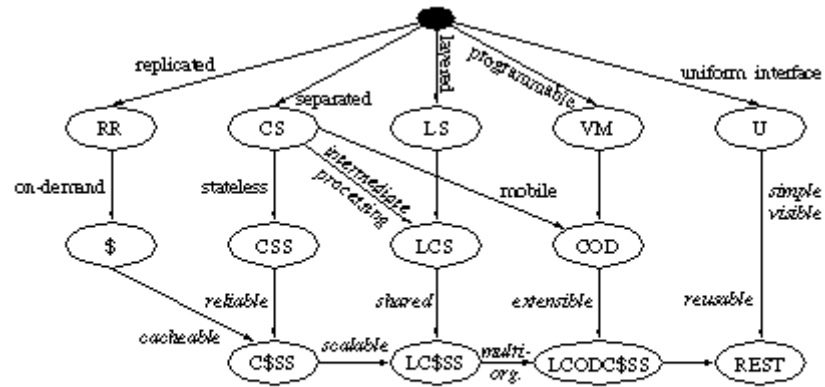


Figure 5-9. REST Derivation by Style Constraints

Statelessness

- Obviously, there is some sense of stateful interaction
 - Browser histories and the back button
 - But that's client side info
 - And order is (more or less) arbitrary
 - The *interaction* in question is the HTTP request/response
 - Cookies
 - Compare with web forms with hidden fields, or
 - URIs with session info, or
 - Typical search result pages
- Stateful interaction ties client and server (or can)
 - Typically the connection is maintained
 - Reconnect recovery is expensive

Uniform Interface

- The constraints on component interfaces:
 - identification of resources
 - resource manipulation via "representations"
 - self-descriptive messages
 - hypermedia is the engine of application state
- We can see this on the Web:
 - URIs
 - HTTP (GET, PUT, POST, DELETE)
 - HTTP requests and responses
 - HTML (esp. the `<a>` tag)

REST Elements

- Data elements
 - Resources, resource identifiers, representations, metadata
- Connectors
 - Clients, servers, caches, resolvers
- Components
 - User agent, origin server, proxy, gateways

What The Heck Is A Resource?

- "We do not limit the scope of what might be a resource. The term "resource" is used in a general sense for whatever might be identified by a URI. It is conventional on the hypertext Web to describe Web pages, images, product catalogs, etc. as "resources". The distinguishing characteristic of these resources is that all of their essential characteristics can be conveyed in a message. We identify this set as "information resources."-- [WebArch](#)
- "The key abstraction of information in REST is a resource. Any information that can be named can be a resource: a document or image, a temporal service (e.g. "today's weather in Los Angeles"), a collection of other resources, a non-virtual object (e.g. a person), and so on. In other words, any concept that might be the target of an author's hypertext reference must fit within the definition of a resource. A resource is a conceptual mapping to a set of entities, not the entity that corresponds to the mapping at any particular point in time."-- [Fielding](#)

"More Precisely..."

- "...a resource R is a temporally varying membership function $MR(t)$, which for time t maps to a set of entities, or values, which are equivalent. The values in the set may be *resource representations* and/or *resource identifiers*." -- [Fielding](#)
- "By design a URI identifies one resource." -- [WebArch](#)
- Anyone notice the vacuity?

Representations

- Why a resource/representation split?
 - Content negotiation (conneg) and generated content
 - HTML and PDF *versions* of these slides
 - JPEG or PNG or GIF or...
 - Statelessness
 - Representations (can) go stale
 - If you have the thing itself, it cannot
 - Modeling
- Why not?
 - It gets a bit metaphysical

Representation Manipulation

- HTTP core [methods](#)
 - GET
 - (Safely) Retrieve a rep
 - PUT
 - Update the resource
 - POST
 - Make a new "subordinate" resource
 - DELETE
 - "Delete" the resource
- GET and POST historically dominate
 - GET safety critical
- Where do application specifics go?
 - In the URI, messages, and server logic

Reading & References

- [HTML Design Principles](#)
- [An Essay on W3C's Design Principles](#)
- [Metcalf's Law and Legacy](#)
- [Google; AT&T shocked by iPhone usage](#)
- [HTTP/1.1](#)