I'll See You on
the Write Side of the Web
WS-REST 2011
Introduction

• **Stuart Charlton (@svrc)**

• Director at Canadian Pacific Railway

• Formerly CTO of Elastra, a cloud computing product based on semantic web technology

• Weblog: *Stu Says Stuff*
  http://www.stucharlton.com/blog

• Many thanks to commenters and twitterers on this topic
Theme

• The Web Architecture has been an immense success...

• ... and yet, we can do better.

• There’s a need to design the software for the **write side** of the web to **scale** and become nearly as **serendipitous** as the **read side**

• Is this even possible? Let’s find out.
The Read Side

- GET
- RDFa/Microformats
- Browsing
- Atom & RSS feeds
- Search
- Semantic Web
The Write Side

- POST
- AtomPub
- Integration
- Facebook Status
- Media Sharing
- e-Commerce
Why?
Why?

- Growth in Centralized Services
- e.g. Facebook
Why?

- Systems Integration
- Custom media types are the current approach...
- ... but that can only be a transitory solution
- Many “RESTful” design thrashing due to lack of prescriptive guidance
- Would be reduced with more generic media types (e.g. as with HTML, AtomPub)
Why?

• REST is not CRUD (create, read, update, delete)
• Neither is HTTP
• POST does not map directly to ‘create’
• CRUD leads to complexity at scale
Why?

• Programming models matter

• In particular, the client’s model of how it interacts with the server

• Process-driven? Or something else?

• Lots of innovation in this space...
Theses

• The Web architecture’s core strength is in encouraging small pieces of independent agreement to be linked together and shared; we’re missing some agreements for writes.

• The Web architecture encourages clients to be designed as agents in a dynamic information space.

• There are practical approaches to programming agents in a dynamic environment.

• It should be possible to create a general purpose media type for systems to manipulate state on the web, in lieu of more specific media types.
Agreement
Collaborative Systems Architecture

- “The greatest leverage in system architecting is at the interfaces. The greatest dangers are also at the interfaces.”

- “When the components of a system are highly independent, operationally and managerially, the architecture of the system is the interfaces.” (Maier & Rechtin, The Art of Systems Architecting)

- In Roy Speak....

- “[REST’s goals are] achieved by placing constraints on connector semantics where other styles have focused on components semantics.”
Design for Serendipity

- “Chance encounters”
- Media types, link relations, RDFa/microformats, URI templates, well-known URIs, host meta, etc.
What agreements could be helpful?

- Link relations for POST resources
- The effects of a POST
  - cache invalidation
  - pre/post conditions
- The contents of a POST
  - e.g. RDF Forms
Programming
Client/Server Programming

- Create Application Logic & Exception Handling
- New Message
- Remote Procedure
- Create
- Handler
- Handler
- New Message
- (optional) New Message
- Request Message Format
- Response Message Format
- Invoke
REST Raises the Level of Abstraction

- The message vs. the resource/representation

- Traditional Client/Server: Client is a program sending/receiving messages

- REST: Client is an agent acting in an information space
Hypermedia Programming

Hypermedia Agent

Goals & Preferences

Cached Representations

Representation Logic
e.g. Link relations, Media type specifications, pre/post conditions

State of the application now

Choose Desired State

Transfer Desired State

Runtime Events

Sensors

Effectors

Modify Goals

Exception Handlers

Resource

HTTP GET

Resource

HTTP POST

Resource

Resource

Environment (The Web)
Qualities of the Client

• Goal-Directed

• Reactive

• Hypermedia workspace
  (cached representations)

• Sensing can be done to pick up on effects
Agents
Agent

• Traditional agents are distinguished from mere programs via...

• The existence of an environment it needs to react to

• The autonomy for the agent to make detailed decisions for the user so long as it is seeking to achieve a goal
The AI approach to Agents

- **Automated Planning**
  - A process to determine an order of actions to be taken in pursuit of a goal

Current state of the world
Description of actions
Goals and constraints

\[ \rightarrow \text{Planner} \rightarrow \text{Required actions (plan)} \]
Example of Planning

Set of all available actions

Plan (selected and ordered actions)
An Alternative Approach

- Subsumption Architecture; aka. Hierarchical State Machines
Programming by Difference

• Coined by Miro Samek

• State nesting lets you define a new state rapidly in terms of an old one, by reusing semantics from the parent

• **Reuse** what is common, **override** what is not
States in the Halo 2 Game Engine
Agreements as a State Sandwich
Media Type
Mapping Nested State Transitions to Restbucks

Goal Hierarchy

- Strategic Goals

Role of Layer

- Abstract Protocol
  - Buy Coffee

- Protocol Binding
  - Sense Coffee Menu
    - Order Coffee
    - Pay for Coffee
  
- Transfer Protocol
  - GET Menu
  - POST Order
  - PUT Payment
  - POST Confirm
State Chart XML

• Promising W3C Draft; Hierarchical FSMs; JavaScript Code-on-Demand, Expressions (Pre/Post Conditions), Event Model

• Weaknesses: Lacks Hyperlinks, Mandates Heavyweight Message Format

```xml
<state id="S" initial="s1">
  <state id="s1" initial="s11">
    <onexit>
      <log expr="'leaving s1'"/>
    </onexit>

    <state id="s11">
      <onexit>
        <log expr="'leaving s11'"/>
      </onexit>
    </state>
  </state>

  <transition event="e" target="s21">
    <log expr="'executing transition'"/>
  </transition>
</state>
```
Possible Characteristics

• JSON-based (and/or JS code on demand)

• Link relations for states, events, and transitions

• Events become identifiable elements of some representations
Behave: A JavaScript State-Aware User Agent

- Implemented in node.js
- JSON-based linked state machines
- First release in May 2011
Revisiting the Theses

- **Agreements**: The effects of a POST in context to other representations

- **Programming**: Instead of a RESTful client library, aim for an agent runtime

- **Agents**: Hierarchical state machines are promising today; hierarchical planning with sensing is a promising thread for research

- **Media Type**: Link relations for states and their transitions; the ability to nest states & transitions via hyperlinks