

Applications of the REST Principle



Liming Zhu

National ICT Australia (NICTA), ²LIXI

Liming.Zhu@nicta.com.au

July, 2008



Australian Government
Department of Broadband, Communications
and the Digital Economy
Australian Research Council

NICTA Members



Department of State and
Regional Development



NICTA Partners

- REpresentational State Transfer (REST) and Resource Oriented Architecture (ROA)
 - REST/ROA Myths
- Applications of REST and ROA
 - Basic web publishing
 - Data-intensive integration (Micro-formats and Mashups)
 - Process-intensive integration (RESTful workflows)
- Applying REST/ROA in Australian Lending Industry
- Conclusion

About NICTA (National ICT Australia)



- Established 2002
- Funded by government and partner Uni.
- 421 research staff, 293 postgrad students
- Our group: Managing Complexity (software engineering)
 - Business and software processes
 - Software architectures



Understanding REST



- Representational State Transfer (REST)
- REST is an *architectural style* for distributed systems
 - Additional constraints on existing client-server style
 - Scalability, evolvability, interoperability and visibility
 - Responsible for the success of the Web (HTTP1.1)
- From to HTTP/WWW to “RESTful services”
 - Alternative/complimentary approach to “Big” Web services
 - Use the “Web” **correctly** and **programmably**
 - Examples: *Amazon, Google...*

REST Principles (1/2)



- P1: Resource is the key information abstraction, identified by a URI
→ **More visibility to business data!**
- P2: A resource may have representations which indicate the state of the resource and can point to other resources through links.
Hypermedia is the engine of application state.
→ **Better business content negotiation and dynamic contract!**
- P3: All interactions are context-free
→ **Better system scalability and simplicity!**

REST Principles (2/2)



- P4: Only use a few primitive operations
→ **Better business interoperability!**
- P5: Idempotent operations and representation metadata are encouraged
→ **Better system scalability and performance!**
- P6: The presence of intermediaries is promoted
→ **Opportunities for new value added services!**
- **Resource Oriented Architecture (ROA)**
 - Resource centric view: Noun-ify verbs → paradigm shift (difficult)
 - Scoping information must be in the URI (not payload)
 - Connectedness: Links are not *just* data relationships

- Just for CRUD, not business logics?
- No description languages? (WSDL, WADL, XML Schema)
- Too much information exposure?
- Only works with HTTP?
- No tool support, no design guidelines?
- No Pub/Sub and Notification?
- No asynchronous interactions?
- No transactions?
- No reliability?

Just myths! You can overcome all of them.

- Using WWW/HTTP does not mean REST compliant!
 - WWW/HTTP abuse
 - Use GET for everything
 - Use POST for everything
 - Ignore cache, MIME types and ETags (remember the REST principles)
 - Underuse response codes and hypermedia/links
 - Misuse URLs for encoding operations, not as resource identifiers
 - Mismatch semantics between the HTTP verbs and applications
 - Same symptom for many APIs (Flicker..) and “platforms” (Facebook..)
- Should
 - Model fine-grained (not whole documents) resources with URL identifiers
 - Dynamic resolution to system identifiers
 - Use links not just for data relationships
 - Metadata, metadata, metadata!

REST for Data-intensive Integration



- Web 2.0 & Semantic Web: different approaches, same goal
 - Web 2.0: not very RESTful but can be.
 - usability driven; limited semantics; mash-up requires manual tuning
 - Future: Computational REST (How to use mobile code (AJAX), RESTfully)
 - Semantic Web:
 - Semantic alignment of data; lack of agreed annotation/access methods
 - Future: RDFa?
- Should: leverage existing RESTful technologies
 - “Feed” technologies: Canonical application of REST
 - Internet Engineering Task Force (IETF) ATOM/APP
 - Extensible: e.g. Google GData API
 - Limitation: How to fit any XML schema into it?
 - WebDAV
 - Semantic RESTful services

REST for Process-intensive Integration



- Still under research, but promising
- Process concepts (case, state, task) are modelled as resources
- Transitions are modelled as hyperlinks or URI templates
 - Micro-formats based
 - “Subjunctive” actions (hypothetical tasks)
 - Dynamic process contract
- Mobile workflow for process execution
 - CREST (Computational REST) principles
 - Mobile workflow beyond AJAX/mobile code
- Clients have local decisions on process execution
 - Based on local information

Example: REST for Process-intensive Integration



```
PUT /jobapplication/appsubmission/1 HTTP/1.1
Accept: application/xml
HTTP/1.1 201 CREATED
Date: May 6 2008 13:32:08 GMT
Content-Type: application/xml
<JobApplication>
  <candidate>Ronnie</candidate>
  <applicationID>1</applicationID>
  <state>Submitted</state>
  <nextsteps>
    <task>
      <role>Recruiter</role>
      <URL>http://localhost:8182/jobapplication/appreview</URL>
    </task>
    <task>
      <role>Candidate</role>
      <URL>http://localhost:8182/jobapplication/apprejectionbycandidate</URL>
    </task>
  </nextsteps>
</JobApplication>
```

CASE: RESTful Architecture for Australian Lending Industry



- LIXI (Lending Industry XML Initiative)
 - Australian Lending Industry
- LIXI goals:
 - Vocabularies (schema); Conversations (process)
 - Reference architectures and implementations
 - Big Web Services, BPEL?
 - RESTful?
- Joint LIXI – NICTA Project
 - *“Visible loans” focuses on publishing loan data, RESTfully*



Motivation: Good Fit with REST



- Reduce infrastructure cost
 - No heavyweight Web services and middleware
- Make data publishing/consumption *simple* and *timely*
 - Accommodate different data consumption capabilities
 - Allow intermediaries to add value (mash-up)
- Promote technology-level interoperability
 - Mass interoperability
 - Reduce cost of change for pair-wise connections
- Create a federated business-technical model
 - Allow autonomy while promoting cooperation
 - Accommodate different planned and un-planned uses

Analogy to Other Federated Models



	Social Networking	Lending Industry
Players	MySpace, Facebook, LinkedIn, Plaxo, Salesforce... 	Lenders, Brokers, Insurers, Aggregators...
Federated business-technical model	Google OpenSocial Specification 	LIXI Visible Loan Specification (Ref. Arch.&Impl.)
Functions	Distribute people info + Pub/Sub + Persistence	Distribute loan info + Pub/Sub + Adaptation
Technologies	GData (extension of ATOM) RESTful API just proposed	RESTful API + LIXI extension of ATOM
Future	Data mash-up, “gadget” hosting, info. collection	Process mash-up, richer interfaces, info. sharing

- Reference Architecture documented in “*LIXI Visible Loans – a Pub-Sub based Service*”
 - How to **publish structured LIXI data** using REpresentational State Transfer (RESTful) services
 - Create finer-grained exposure, not just full documents
 - Use links judiciously
 - Promote intermediaries through extensive use of metadata facilities
 - How to **publish and subscribe to change notification** using ATOM Publishing Protocol (APP)
 - Use APP extension to embed LIXI micro-formats
- Implementation
 - Server side: Loan information publishing server
 - Native XML database + REST/APP based publishing module
 - Client side: Client library + Excel plug-in

Conclusion



- REST-compliant needs very conscious effort
- Great benefits in a good RESTful design
 - Inherit WWW infrastructure
 - Inherit WWW quality
- RESTful Data publishing is quite mature
 - Don't abuse WWW/HTTP
 - From APP to microformats to RDFa
- RESTful processes (workflow) design still a challenge
 - CREST + Process mashup