A Web Services Manifesto

Phillip J. Windley
phil@windley.org
www.windley.com

The Williams Family

Life Event: *Moving to Utah*

- Change of address
- Register car
- Register to vote
- Enroll child in school
- Bussing
- City services
- Health information
- Child safety
- Check the commute
- Tax information

Federating Services

Moving to Utah

- Real estate
- Taxes
- Register car
- Register to vote
- Enroll child in school
- Bussing
- City services
- Utilities
- Health information
- Banking
- Child safety
- Change of address
- Check commute

Child entering School

- Health information
- Grades
- Tuition and fees
- Books
- Child safety
- Bussing
- Federal programs
- Check commute

Shared services

Private services

Web Services

- Start web services now:
  - Incrementally expose your data
  - Incrementally expose your APIs
- Small marginal cost
- The more APIs and data you expose the greater the potential interoperability
- Small, scripted aggregations lead to serendipitous applications

Legacy Data

- Governments and other organizations control vast data resources
- That data is held hostage in disconnected, legacy data resources
- eGovernment requires that we free data from siloed systems and legacy platforms

Enabling Web Services

- There are things we can do, for almost nothing, as we put services online that will have a **huge** impact on future development.
- But…we have to design wisely.

Design Principles

1. *Every* data element and collection is a resource
2. *Every* resource should have a URI
3. Cool URI's *don't* change
4. Preserve the structure of data until the *last possible moment* (*i.e.* return XML)
5. Make XML Schemas available *online* for your XML
6. Data queries on existing resources should be done with a GET
7. Use POST to create new resources

Design Principles (cont)

8. Document your service API using WSDL, WRDL, or some other standard
9. Advertise the presence of the data using WSIL
10. Adhere to data standards such as RSS where available
11. Use Metadata (RDF) for XML
12. Use HTTP authentication as much as possible
13. Make data available in multiple flavors

Web Services

- Web services are self-contained pieces of code with three distinguishing properties:
  1. Communicate in an interoperable XML protocol, such as SOAP.
  2. Describe themselves in an interoperable XML meta-format, such as WSDL.
  3. Federate globally through XML based registry services, such as UDDI.
- Not defined in terms of SOAP, WSDL, and UDDI.

Using Web Services

Examples

- Common Payment Gateway
- Professional Licensing
- Criminal Justice Network (CAD)

The Future: ALIN

- Application Layer Internetworking
- Level seven switching
- Key idea: exposed APIs allow data monitoring and modification midstream

ALIN Applications

- Service call switching
- Context sensitive filtering
- Event monitoring
- Logging
- Service facades
- Message store and forward
- Business rules repository

Why ALIN?

- Separation of concerns
- Reliability
- Access

A Word of Warning

- A good developer should do everything they can to avoid serialization.
- When serialization cannot be avoided, it can be mitigated through caching in some cases.
- Web services is nothing *but* serialization.
- Further, SOAP over HTTP makes caching difficult (uses POST).

Summary

- Remember
  - Forget the hype
  - Don’t try to figure it all out first
  - Jump in and do something
- The keys are
  - XML
  - Incrementally exposing data and APIs

For More Information

- phil@windley.org
- http://www.windley.com
- My paper at above address
- http://www.soaplite.com
- http://xml.apache.org/axis
- http://www.xml.gov