REST Web Services with java

-Prakash Badhe

-Mail:prakash.badhe@vishwasoft.in

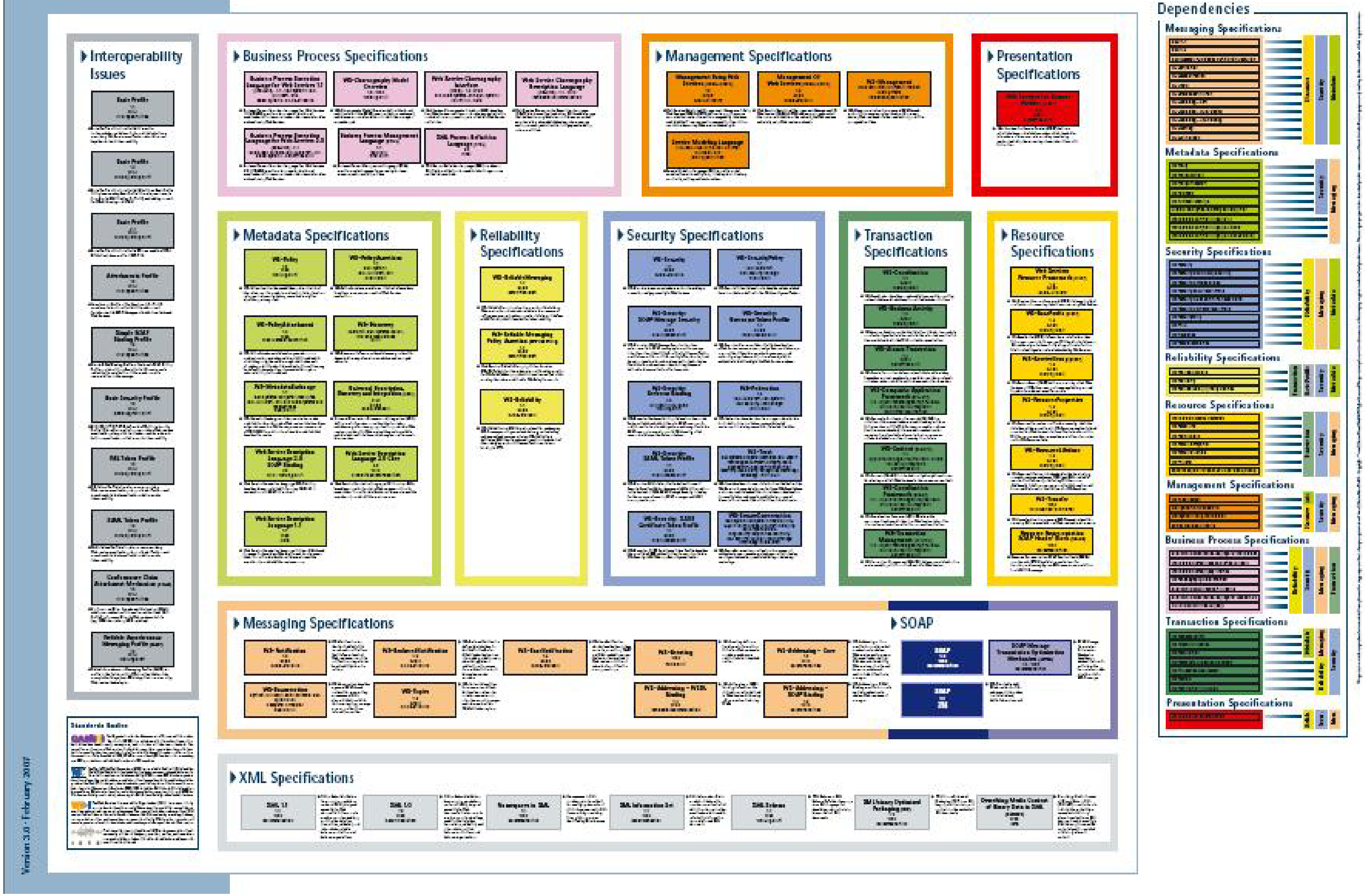
Soap Web Services

- Manage language independent interactions across different platforms and applications
- Different versions evolved one by one to support additional features.
- Interoperability is still an issue across different implementations and versions.
- Heavy dependencies on the soap library and other xml implementations.
- Non-xml data format support is limited only in the form of attachments.
- Performance is an issue for mission critical applications.

What is wrong with SOAP Services ?

- Language first mindset
- WSDL imposes tightly coupled dependencies between services and consumers
- No support for dynamic and functional languages
- With a few exceptions: WS protocols are not ready for the internet
- We have too many WS protocols and too many versions

Web Services Standards Overview


Interoperability Issues
Business Process Specifications
Management Specifications
Presentation Specifications
Metadata Specifications
Reliability Specifications
Security Specifications
Transaction Specifications
Resource Specifications
Messaging Specifications
SOAP
XML Specifications
Standards Bodies
Dependencies
Messaging Specifications
Metadata Specifications
Security Specifications
Reliability Specifications
Resource Specifications
Management Specifications
Business Process Specifications
Transaction Specifications
Presentation Specifications

Version 3.0 · February 2007

RESTing the SOAP ?

Enter 'REST' Services

- REST is 'Representational State Transfer'..
- REST is an architectural style rather than a protocol which removes the dependencies on soap and xml standards as wsdl,uddi etc.
- REST supports any understandable data formats across applications.
- REST works currently only with Http.
- REST specifies Resources/data on server rather than actions on them..
- REST specifies transfer of the state of Resource across applications.

REST Principles

Resource based (not service based)

Addressability (name everything that matters)

Statelessness (no stateful messages exchanged with a resource)

Relationships (expressed through links)

State Machine interactions (not business processes)

HTTP based

What is Resource ?

- A resource is something “interesting” in the system
- Can be anything
  - Spreadsheet (or one of its cells)
  - Blog posting
  - Printer
  - Winning lottery numbers
  - A transaction
  - Others?
- Making your system Web-friendly increases its surface area
  - You expose many **resources**, rather than fewer *endpoints*

Resources on the Web

Resources are not only represented as XML

- XML formats (HTML, XHTML, RSS, etc.)
- JPG, GIF, PNG
- MP3, WAV, OGG
- Anything else that can be on the web.

Resource Nouns

- Important ‘things’ (nouns) are Resources
  - Addressed through a URI
- Uniform interface (verbs)
  - In HTTP: GET, PUT, POST, DELETE
- Verb-noun seperation makes integration easier
  - GET /customer/45 Instead of getCustomer(45)

REST verbs and nouns

- REST works with resources as nouns with their identities.
- The state of these nouns is shared with clients over http methods
- The http methods are specified as verbs in REST
  - Get: get the resource state/values
  - Post : post new resource
  - Put : update the resource state
  - Delete :delete the resource on server
  - Options: read the options available with server.
  - Head : set the http headers on server.

REST Commandments

- Give every “thing” an ID
- Link things together
- Use standard methods
- Communicate statelessly

Addressability of resources

- Resources MUST be represented by URIs
- Retrieve a representation of a resource: GET
- Get metadata about an existing resource: HEAD
- Create a new resource: PUT to a new URI, or POST to an existing URI
- Modify an existing resource: PUT to an existing URI
- Delete an existing resource: DELETE
- See which of the verbs the resource understands: OPTIONS

Resource Links

- Resources contain links (or URI templates) to other resources
- Links act as state transitions
- Think of resources as states in a state machine
- And links as state transitions
- Application (conversation) state is captured in terms of these states

Statelessness

- The interactions with resources are stateless
- Resource state is always kept in the server and sent to the client as representations
- Application state is always kept in the client and is used to modify the state of resources
- Statelessness increments scalability (self contained messages) ..see the overhead of managing Stateful EJB..
- Avoids putting resources in inconsistent state

State management

- REST mandates that state be either turned into resource state, or kept on the client.
- Reduces burden on the server..increases scalability on the server
- Reduces coupling with a specific machine

Http based

- REST principles are not HTTP dependent
- But in practice now REST is be considered an HTTP dependent architecture style
- REST expresses dependencies on HTTP specific concepts such as verbs, URIs and headers
- In the future....non HTTP-based REST will come up..

Need of REST

- We need a natural way to model resource-based services
- We should leverage the principles of the web on SOA applications

REST vs SOAP Battle

Java support for REST

- Jax-RS is the new java specification standard for implementing and consuming REST web services.
- Implementations a are Jboss RestEasy,Jersey platforms.

SOAP and REST

- REST is ready for the enterprise
- REST is strong at:
  - Internet scale computing
  - High levels of interoperability
  - Resource Oriented operations
- SOAP/WS is strong at:
  - Complex security (Trust and Federation)
  - Multi-transport services
  - Occasionally connected applications
- SO scenarios in the real world are typically enabled by a combination of WS and REST

REST for Mobile clients

- By using REST over HTTP instead of SOAP, we can drastically reduce the overhead of message processing
- An HTTP server implementation is feasible for a mobile device

  http://opensource.nokia.com/projects/mobile-web-server/

REST on the WEB

- Google
  - GData, OpenSocial
- Standards
  - Atom, WebDAV
- Amazon
  - S3, SimpleDB
- Microsoft (!)
  - Project Astoria, Web3S
- Um… the Web.

Resources on Social Web

- Users consume content by a lot of sources, in many formats. (Web, Feeds, YouTube, Amazon, Flickr, etc.)
- Users share things they find interesting by exposing a Feed, through Google Reader, Facebook, and FriendFeed.
- The people who read the feeds may share the content with their contacts as well.

ATOM protocol and resource

- Atom Syndication Format
  - A Feed specification based on RSS 2.0

- Atom Publishing Protocol
  - a simple HTTP-based protocol for creating and updating web resources.

Google OpenSocial

- Open Competitor to Facebook’s Application Platform
- Exposes Three RESTful APIs:
  - People and Friends data API
  - Activities data API
  - Persistence data API

Constraints for the REST

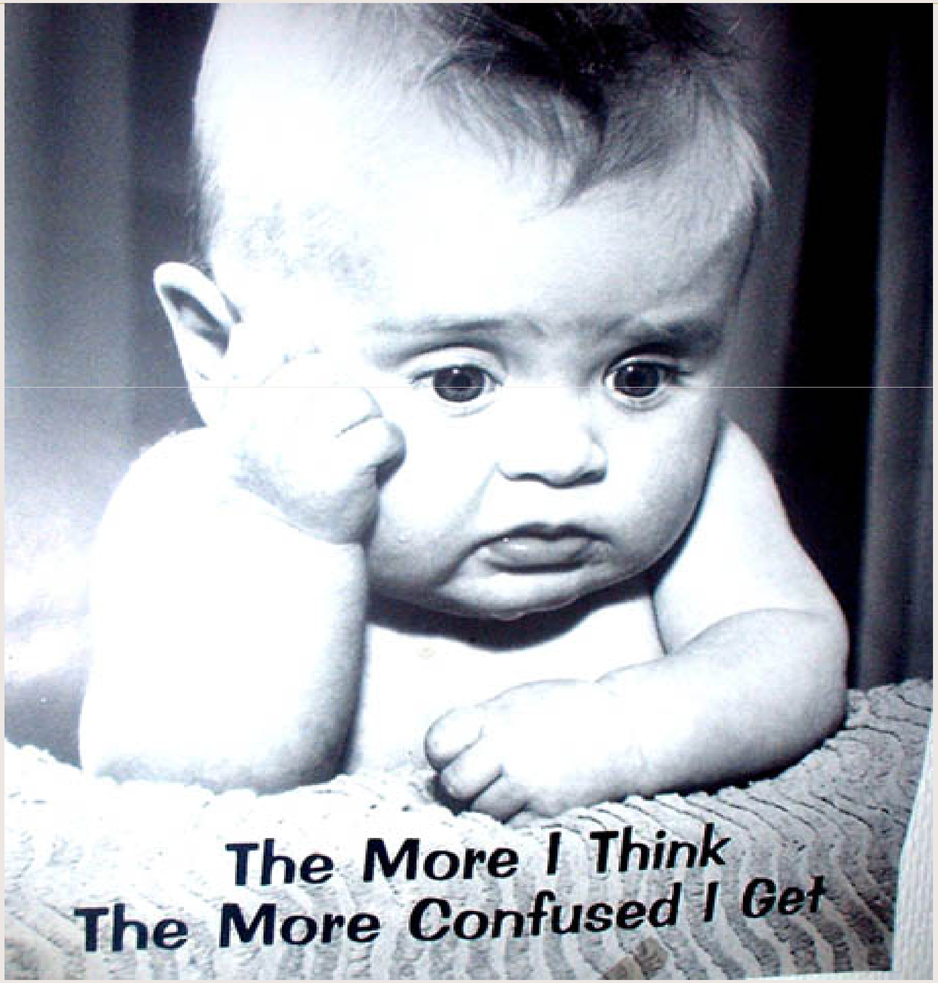
REST has four architectural constraints:

- separation of resource from representation,
- uniform interface,
- self-descriptive messages, and
- hypermedia as the engine of application state.

Resource Oriented Architecture

- ROA is the term for REST on HTTP/URI
- A Service consists of all the resources available within a certain domain of control
- Since REST is a type of SOA, ROA is an implementation of SOA as well.

Questions ?



Thank You!