

# Introduction to Service Oriented Architecture (SOA)

---

Hari Rajagopal  
Galileo International





# Agenda

---

- Definitions
- Background
- SOA principles
- Case study
- Summary





# Component

---

- Reusable unit of code, typically developed for a specific project or app that has been packaged in a way that it can be reused
- Typically a POJO (plain old Java object)
- Deals with the basic business objects





# What Is a Service ?

---

- More generic than a 'web service' !
  - Although that is how the majority are deployed
  
- A well defined unit of software that
  - Performs a specific function
  - Complete in itself
  - Consistent interface





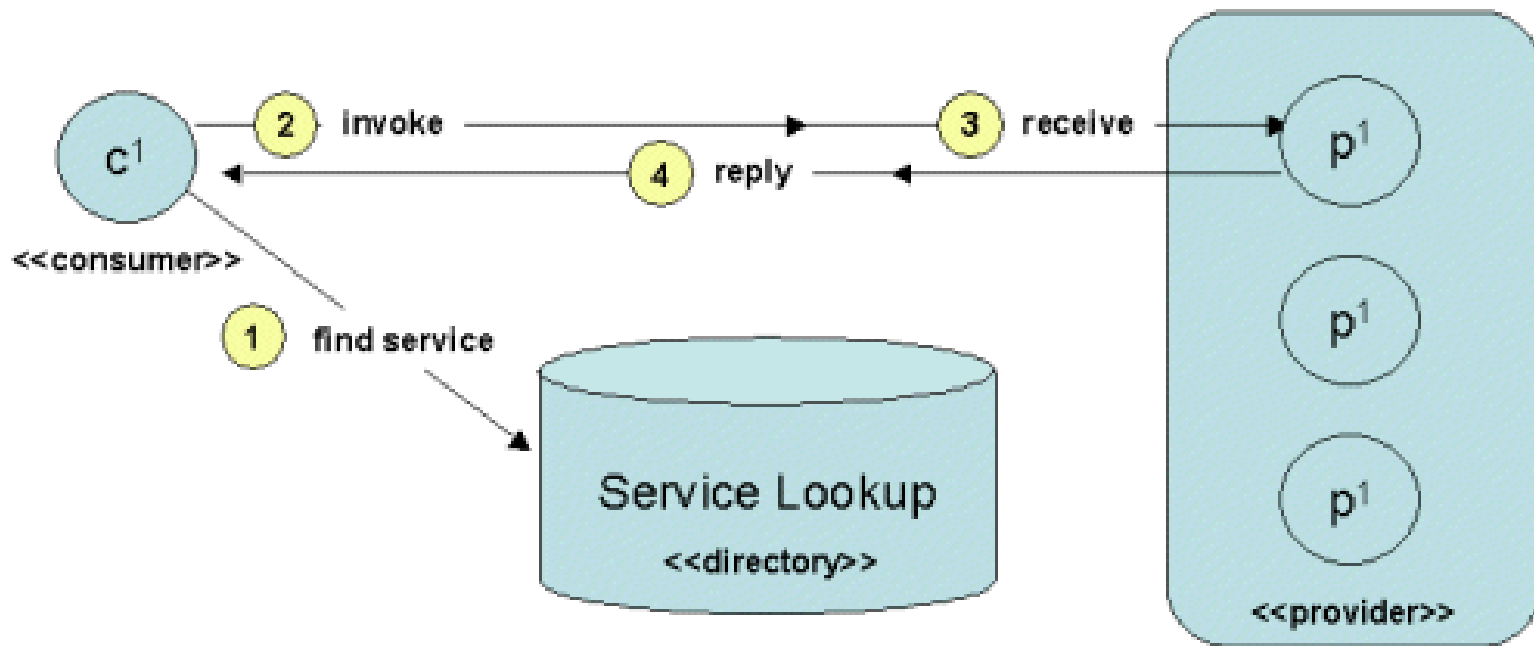
# Classic View of a Service

---

- Provided by a *'provider'*
- Exposed *via a 'directory'*
- Looked up and located by a *'consumer'*



# View of a Service





# Service Discovery...

---

- Has not lived up to its promise
  - XML is not 100% machine understandable
  - RDF descriptions may be the key??
    - (Paul Giangarra devotes a section of his talk to this)
  
- Services are still 'handed' to clients with documentation, API support groups handle issues





# Enterprise Service Bus

---

- Is a standards based messaging platform that provides XML transforms, content based routing
- Can be used as the basis for implementing the transport layer of SOA

 Offerings from IBM, Fiorano, Soniq, ...



# ESB Usage Patterns

---

- Router
- Broker
- Adapter
- transformer





# Router

---

- Content based
  - Typically introspects into the message and determines target endpoint
  - Configurable using XSLT expressions
  - Used for routing based on customer and/or framework cues inserted in the message stream
    - Another case for keeping the pattern of interception active within the framework





# Service Oriented Architecture

---

- An architecture that emphasizes:
  - Coupling of existing and new architectures seamlessly
    - Moves components out of silos into the mainstream
  - Responsive and agile architecture
    - Easy to rearrange and reassemble components





# Where Did It Come From ?

---

- SOA has been around in various guises
  - Implementation has become easier due to the advent of WS-standards
  
- Some of these may sound familiar
  - Object Brokers (ORB)
  - RPC based brokers
  - Middleware based integration mechanisms



# Is a SOA Simply Web Services?

---

Web Services are one possible means of implementing a SOA

In the past, client-server SOAs have been attempted





# SOA Functionality

---

- Base functionality of a SOA implementation:
  - Deploy services within its domain
  - Integrate the services and make them available
  - Orchestrate the integrated services
    - In a reliable fashion





# SOA Structure

---

- Basic unit of a SOA framework is a service
- Integration usually occurs through open systems protocols (SOAP, JMS )
- Orchestration is accomplished using BPEL4WS and the like





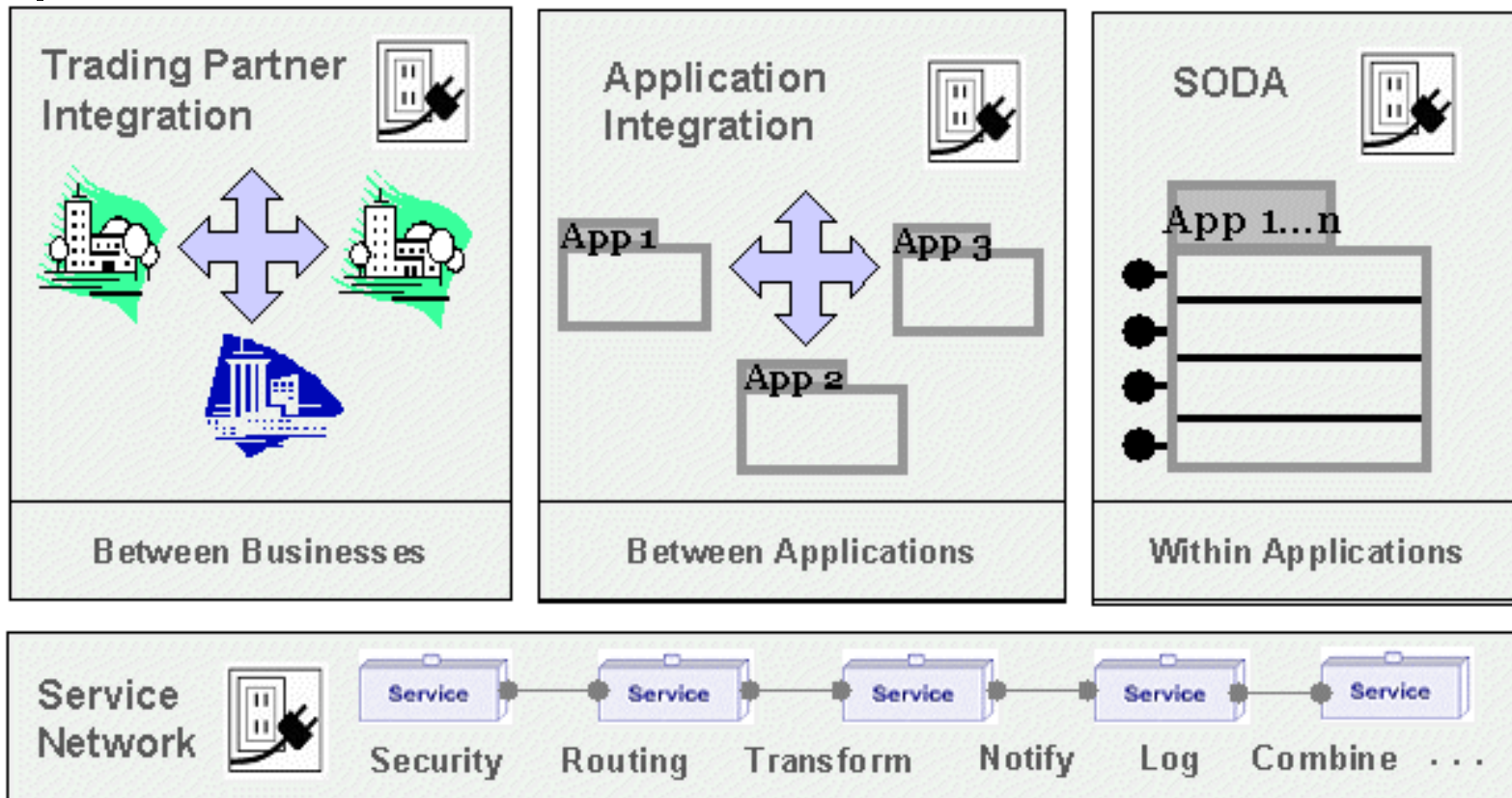
# Service Network

---

- A application level network that leverages and implements a SOA
  - Consists of various interacting applications that are composed of services (facilities) that are hosted within a SOA
  - True enterprise integration and reuse
- Marketing may use same service as KM for charting



# Service Network Boundaries





# What Does a SOA Buy You?

---

- Reusable services
- Applications are composed of *services* rather than monolithic units
- Facilities provided by the SOA framework leave the developer free to code business logic





# What Are the Participants?

---

- Servers (Web and application)
- Directories / registries
- Routers / brokers
- Enterprise Service Bus
- Transaction processing monitors





# Case Study

---





# Acronyms

---

- GDS – Global Distribution Service
- Service provider – in this case airlines, rental car companies, hotel chains
- Vendors – see above, except now we have aggregators in the mix as well





# SOA Implementations

---

The classic 'build versus buy'  
dilemma





# Trade-offs

---

- Build

- Expensive
- In-house expertise needed
- Proprietary implementation
- Lack of input to standards bodies

- Buy

- Expensive (long term)
- In-house expertise needed to develop against
- Tied to vendor implementation



# Open Source Offerings??

---





# Some Travel Company

---

- Primary business focus is the housing and supply of travel related information
  - Air itineraries, hotel reservations, car rentals
- Question:
  - How do we provide this information to different business units?
  - How do we provide services to access this information to clients of the business units?





# Solution

---

- 1<sup>st</sup> phase – expose traditional artifacts as SOAP based web services that are easily available to clients
- 2<sup>nd</sup> phase – house these services within a SOA framework such that they are reusable across the enterprise



A graphic consisting of overlapping colored squares (yellow, red, blue) and a black crosshair.

# Phase 1

---

- Put Web Service front ends on legacy assets such as GDS data
  - These are housed on 'big-iron' mainframes
  - Expose these *via* XML based APIs
  - Provide aggregated (higher level) services for the presentation tier





# Flaws Revealed

---

- Data is still in the legacy format
- Scalability issues
- No way to bill for the transactions
- Versioning and cutover issues





## Which Leads to the Following...

---

- What are some of the facilities provided by a SOA ?
  - Security
  - Service lifetime management
  - Reliability
  - Reporting & logging
  - SLA management





# First Steps in Implementing SOA

---

- Design and develop a messaging core
- Use standard protocols to interface with external entities
- Build with the philosophy of location transparency



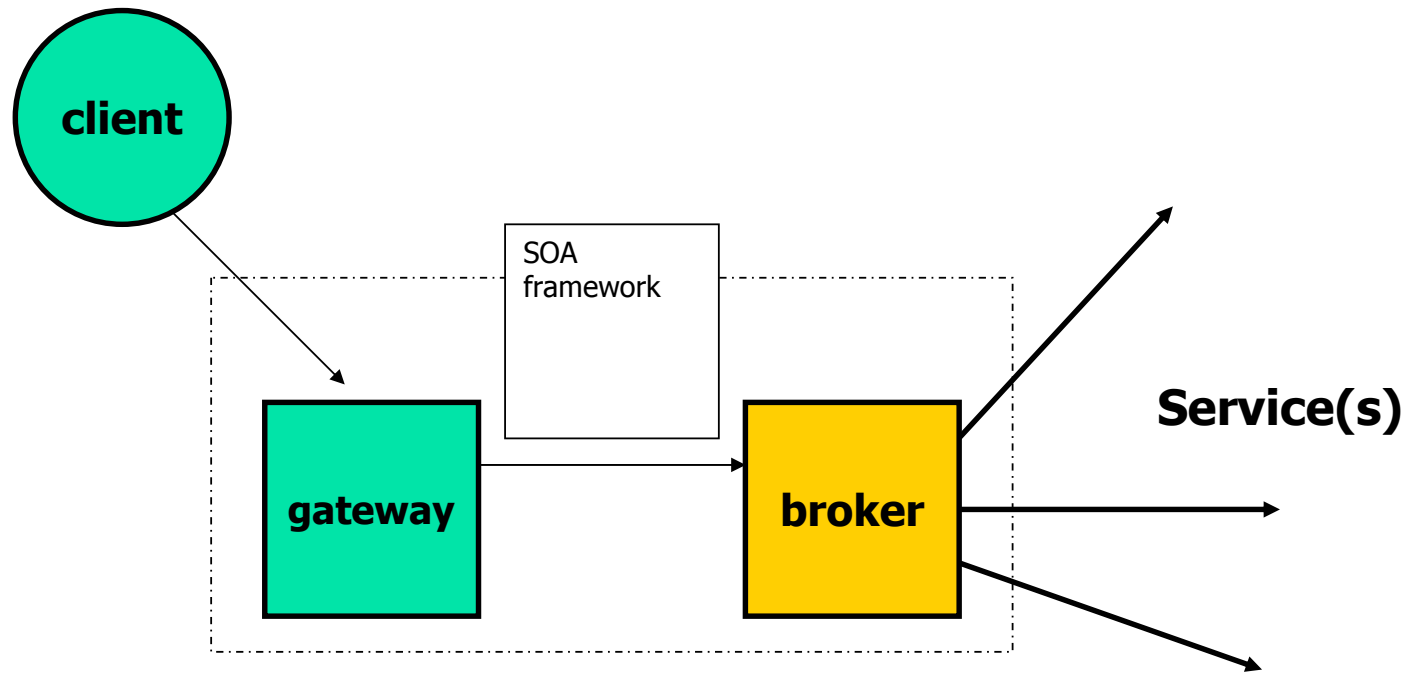


# First Steps in Implementing SOA *(Continued)*

---

- Build an agile core that is merely a message router (much like a *bus*)
  
- In the interests of scalability and security partition the implementation between the presentation and business tier (note: the codebase running on every tier is the same, one simply 'knows' that it is a presentation layer box)

# Architectural Layout





# Building Blocks

---

- Apache Axis
- Fiorano ESB
- Tomcat
- DB2





# SOA Provides Services Too...

---

- Infrastructure services are provided to developers
  - Authentication of clients using the services
  - Logging of transactions
  - Security and DOS attack prevention
  - Load based routing and balancing





# Gateway Tasks

---

- In the previous slide, the gateway layer:
  - Receives messages
  - Verifies authenticity
  - Performs format translation (more on this later)
  - Routes to the appropriate service endpoint



# Gateway Architecture

---

- Servlet based
- Uses filters to intercept and process messages
- Content based decision making ability





# Authentication

---

- In this case since most clients were coming in using HTTPS and port 443 simple Basic authentication was used
- Authentication information is carried in the HTTP headers as encrypted data (base-64)





## After Basic Authentication...

---

- A token is carried out of band with the payload
- This 'call-context' also contains correlation information for reconstituting audit info
  
- The token is re-asserted at various points along the way





# Service Broker

---

- The broker serves as a lookup agent
- It also initializes the services and brings them into play
  - Note: the services are dormant and waiting since the SOA framework startup (lazy init)
- Broker gets service information from the registry





# Registry Information

---

- Service name
- Service version
- Service payload format
  - Input type
  - Output type
  
- Location of the service instance (URL)





# Payload Formats

---

- Object or XML ?
- XML – usable by diverse client platforms
- Object – performance gains
- XML appliances may make this a non-issue





# Transcoding

---

- SOAP – *'lingua franca'* is XML
- Objects work best for performance
- Solution – *'transcode at the edge'*





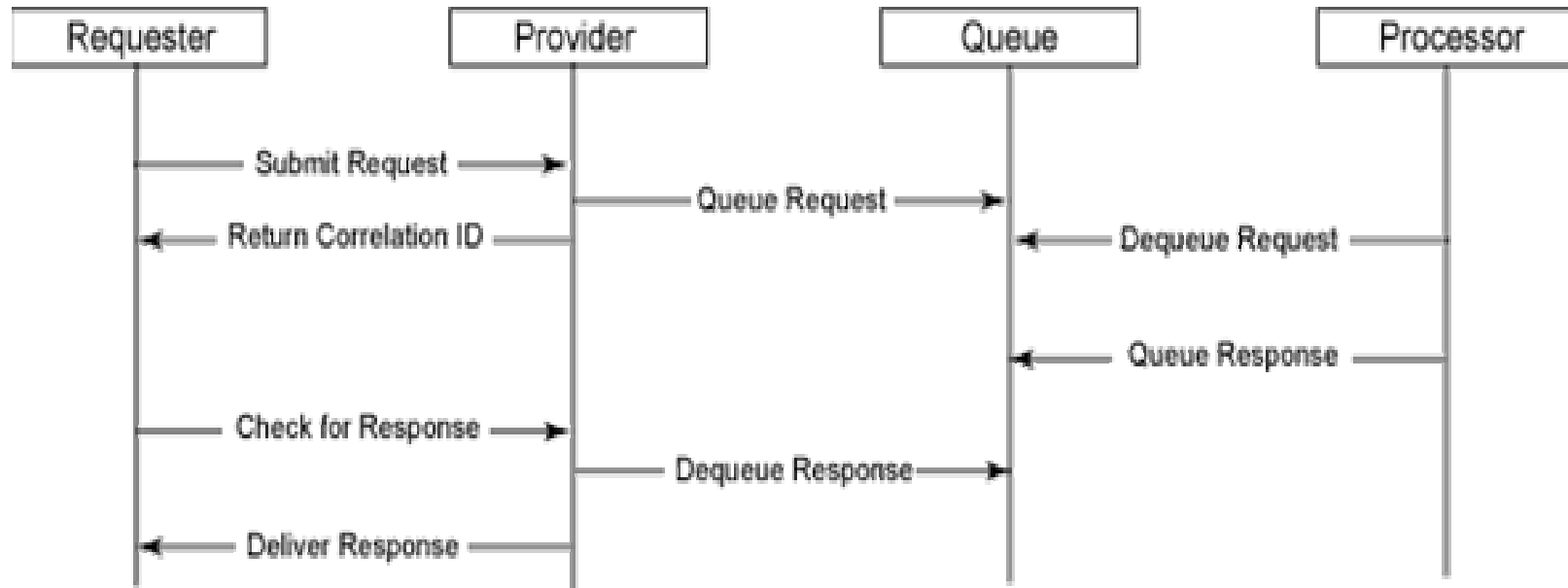
# Long Running Transactions

---

- Typically web service transaction times span more resources than we are used to
- ACID conditions may not be enforceable
- Can we relax any of these?
- Which ones?
  - In some cases consistency is relaxable
  - In others – atomicity



# Asynchronous Messages





# Business Domain Model

---

- A common representation of the assets of the business
- Reusable across domains within diverse development environments
- Language neutral model to base services
- Ease of service interoperability





# How Are Services Used?

---

- Located by name and version
- Accessed by clients that generate interfaces based on WSDL
- WSDL uses XML types that derive from the BDM





# BDM Based Approach

---

- Develop your business model (UML)
- Generate XSD schema from the model
- Compose service interfaces from these business model schemas
- Generate language specific objects from these using binding frameworks





# Routing

---

- Typically, gateways route to brokers within the framework
- This allows
  - Load based routing
  - Transparent routing to upgraded services
  - Handling of version inconsistencies
    - Consider the following example





# Broker to Broker Call

---

- Client calls service A on a Service network
- Gateway authenticates the client
- Gateway determines whether the appropriate service version exists at that node
- Routes to the appropriate node in the domain



## Alternate Scenario

---

- The version of the service requested by the client (based on attribute in the payload) is not found at this node
- Gateway:
  - Looks up service info for that domain
  - Finds an alternate
  - Applies transform as needed (object → XML)
  - Routes to the appropriate node





# Deployment Strategy

---

- A rack of inexpensive bladeservers
- Identical codebase on each blade
- A spare pool of blades
- Addressable under a common Virtual IP





# Summary

---

- Build a micro-broker routing core
- Build in SLA management from the get go
- Use COTS ESBs to build the transport backbone and do your transcoding
- Common business semantics (BDM)





# So, Now That It's Built...

---

\$1 million question is –  
Will they come ?





# Service Level Agreements

---

- A Service Level Agreement (SLA) defines the minimum level of service that a client will tolerate or the business has contracted to provide
  
- Can make or break adoption of your services





# Service Level Objectives

---

- The SLA with a client may be composed of one or more Service Level Objectives (SLO)
- These include:
  - Response time
  - Availability and uptime of service
  - Accuracy of data returned





## Example of a SLA

---

- United airlines contracts with a website provider that presents its fares to the public
  - A customer shall have a mean response time of 5 seconds
  - This shall be averaged over a month
  
  - The accuracy (matching of quoted fares *vs* booked price) shall be within 5%





# Backing Up Your Claims

---

- How do you deliver this level of service ?
  - Manage your application dynamically
  - Monitor it so you can prove response times, accuracy, *etc.* – in case of an audit (CYA)





# Service Level Management

---

- Monitoring
  - Is a service level threshold exceeded ?
  
- Management
  - Taking corrective measures to restore the level of service





# Back to Our Case Study

---

- A COTS – AmberPoint SLM manager
- Custom code to integrate the SLM manager with the SOA framework
- End result – better uptime, faster response





# SLM Approach

---





# Business Process Control

---

- Now that services are built and available, how are they to be reused within the enterprise?
  - Compose 'Meta' services of the more granular services
  - Orchestrate these using a standard mechanism



# In Our Travel Domain Case Study...

---

- Finer grained services would be:
  - Air availability or Flight Information lookup
  - Booking service
  - Travel codes translator
  - Credit card authorization service
  - ...
  - The finer grained services are rarely used directly by web clients – exception being the TCT





# Orchestration of Services

---

- What happens when a vendor changes, inventory is not available or a route to a GDS is unavailable on the network ?
- Business rules defined in BPEL4WS automatically route the service to other resources to satisfy the request





# Meta Service

---

- Trip Planner service
  - Takes trip endpoints and returns an collated response that displays trip choices, different routes, schedules
    - Uses FLIFO to get a list of possible routes and schedules
    - Uses the travel codes translator to decode industry specific codes (DEN, LAX)
    - Uses Destination Resolution services to display value-adds for the destination (golf packages, tours)





# Orchestration

---

- Standards have jelled recently
- BPEL4WS is a joint effort of the following companies
  - IBM
  - Microsoft
  - HP





## Orchestration *(Continued)*

---

- BPEL4WS (the language) focuses on building services that are composed of other services
- The language defines a *process* in terms of a number of *activities* involving interacting *partnerlinks*
- BPEL process is itself a service





# When Should We Use SOA?

---

- Use SOA principles everywhere that flexibility is desired
- ESB on the other hand is NOT a must have in all projects
  - Typically an architecture that is geographically distributed can benefit from this
  - As also – process flows that are very asynchronous





# What Next?

---

- Grids and autonomic computing
- Service network aggregation
- Dynamic discovery and request satisfaction
- ...
- ...
- ??????????????





# Related Presentations

---

- Two other sets of presentations, by
  - Paul Giangarra
  - Paul Freemantle
  
- If this presentation was interesting please check those out.





# References

---

- <http://www.serviceoriented.org>
- <http://www.ibm.com/developerworks>
- <http://builder.com.com/5100-6386-5064520.html>
- <http://www-136.ibm.com/developerworks/webservices>





# My Contact Info

---

- [Grimgaunt@yahoo.com](mailto:Grimgaunt@yahoo.com)
- [Hari.rajagopal@galileo.com](mailto:Hari.rajagopal@galileo.com)

