MEMBERSHIP
ARTICLES
ALL ARTICLES
WHITE PAPERS
ROUND TABLES
PRESENTATIONS
NEWS CLIPPINGS
EVENTS
TRAINING
WORKSHOPS
CONSULTANT NETWORK
SOLUTION LOCATOR
SEARCH

OTHER TOPICS
BPM
BIZ DECISION MGMT
BIZ ARCHITECTURE
ORG. PERFORMANCE
INNOVATION
GOVERNMENT

ARTICLES

Resource Oriented Architecture

By: Jeremy Deane, , Technical Architect, Collaborative Consulting
Wednesday, March 11, 2009

According to a 2008 Gartner Survey[1] there has been an increase in the number of organizations implementing web services using Representational State Transfer (REST) and Plain Old XML (POX). RESTful web services are less complex, require fewer skills and have a lower entry cost than WS-* SOAP web services. However, RESTful web services by themselves do not provide a complete enterprise solution.

Resource Oriented Architecture (ROA) *extends* the REST architectural style and provides a deeper, more extensible and transport independent foundation. While RESTful web services require the use of HTTP, resource-oriented services support additional transports such as JMS or FTP. In this article, I will cover the key concepts of ROA referencing a set of resource-oriented services implemented using 1060 NetKernel[2] and Apache ActiveMQ[3]. The demo, along with the source code and tests, can be downloaded from the Collaborative Consulting download page[4].

Resources are the foundation of ROA; a *resource* is an abstraction of information. For instance, an employee is an abstract set of data points that can have different representational forms such as a XHTML, JSON or XML. Each immutable *resource representation* is identified by a relative Universal Resource Indicator (URI) and may contain links to other resources.

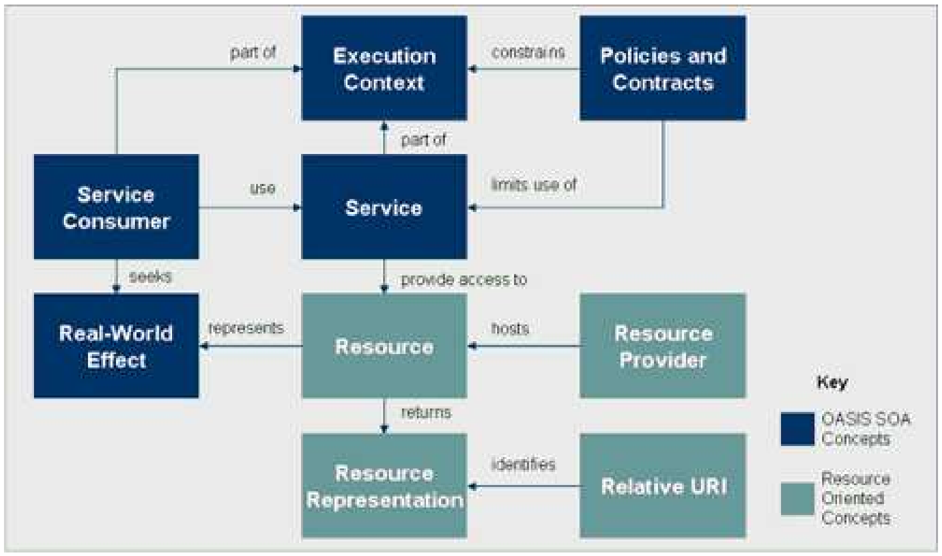


Figure 1 ROA Conceptual Model

A URI consists of an address scheme, authority information and a path:

URI Syntax	*<scheme>:<authority><path>*
URI Example	*host/context/employee/1234*

But a relative URI simply refers to the path or address, '/employee/1234'. Consequently, the resource representation identifier is transport independent because it is divorced of a scheme (recall: *http://* from prior example) and can be reused in multiple environments (e.g. QA or PROD) because it does not contain authority information. Finally, a *resource provider* is used to host a resource (System of Record) and a *service* exposes a uniform set of actions, **C**reate, **R**etrieve, **U**pdate, and **D**elete (C.R.U.D.), for stateless access to a resource.

Since resource-oriented services are transport-independent there must a mechanism for exposing them to the outside world. This is accomplished using *transports* that reside at the edge of a system, detect external events and map those events into internal resource requests. For example, an HTTP transport listens for a consumer's request for a Uniform Resource Locator (URL), specifying an access method (e.g. GET, PUT, POST, & DELETE). The URL is transformed into an internal relative URI and the access method is translated into an action.

SOLUTION LOCATOR

Expedite your research.
Find specific SOA solutions and request information.

ORACLE
ORACLE

SOA WATCH COLUMN

SOA Watch: When Considering Services...

Read this column.

Services are the building blocks of SOA, and like building blocks of a house or a building, the quality will define the value of the finished product. In this case, the SOA itself. Thus, spending...

EXPERTS WANTED

Would you like to:

- Submit an article
- Lead a Round Table
- Speak at a Conference

Contact us today!

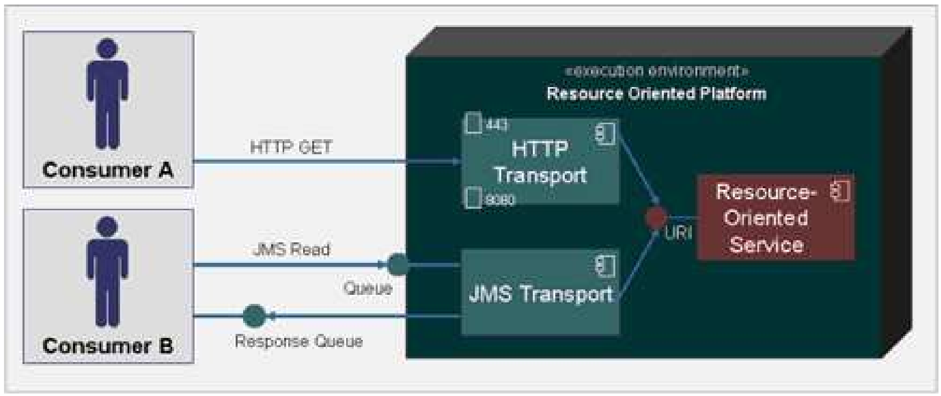


Figure 2 Resource-Oriented Service Example

A JMS transport uses the *properties of the message* to map the request to a resource-oriented service. The two required properties are the relative URI and action; and if the consumer requires a response, as is the case when the action is retrieve, then the *ReplyTo* property is populated. In addition, the message *type* is used to determine if the submitted resource representation is binary or text.

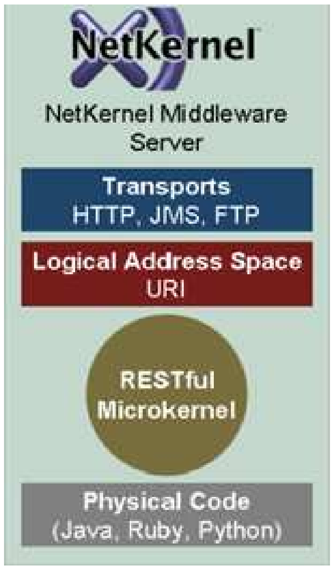
Resource-oriented services use the transport to provide a guarantee of consumer identity or *authentication*. In the case of HTTP, Basic or Digest Authentication[5] is used while encrypted credentials are passed as a property if the transport is JMS. Consumer and provider can insure *confidentiality* by securing the transport (e.g. SSL or VPN) or securing the message using encryption.

To prevent unauthorized resource access consumers are granted privileges or roles (sets of privileges) where a privilege is comprised of a relative URI and an action. Additionally, regular expressions can be used to represent the URI decreasing the number of privileges that are required for *authorization*.

Action	**Relative URI (regex)**	**Description**
Create	/employee/\S{4}	Create an employee
Retrieve	/employee/\S{4}	View employee
Retrieve	/business/.*	View any business resource
Update	/business/department/\w*	Update any department
Delete	/employee/\S{4}	Delete an employee
Delete	/business/department/\w*/employee/S{4}	Delete department employee

And actions that change the state of a resource including, create, update, and delete, are *audited*.

ROA can be implemented using several technology platforms including the Spring Framework[6], Ruby on Rails[7] and NetKernel. However, only NetKernel is built upon a Resource-Oriented Computing Platform[8] (hence, "ROC"). The core of resource-oriented computing is the separation of logical requests for information (resources) from the physical mechanism (code), which delivers the requests. Services built using ROC have proven to be small, simple, flexible and require less code compared to other technology platforms.


NetKernel
NetKernel Middleware Server
Transports HTTP, JMS, FTP
Logical Address Space URI
RESTful Microkernel
Physical Code (Java, Ruby, Python)

NetKernel provides access to resources identified by a Universal Resource Identifier (URI) address. All URI addresses are managed within a logical address space. A REST-based micro-kernel handles all requests for resources, resolving the URI from the address space and returning a representation of that resource. Requests to the micro-kernel can also be made to create new resources or update or delete existing resources.

The demo consists of a set of resource-oriented services used to manage personnel. The employee resource-oriented service allows a consumer to create, retrieve, update and delete resources either via HTTP or JMS. For instance, a consumer can create an employee by issuing an HTTP PUT of an employee representation or by sending an employee representation to an ActiveMQ queue. Both approaches result in an internal sub-request to update a department resource.

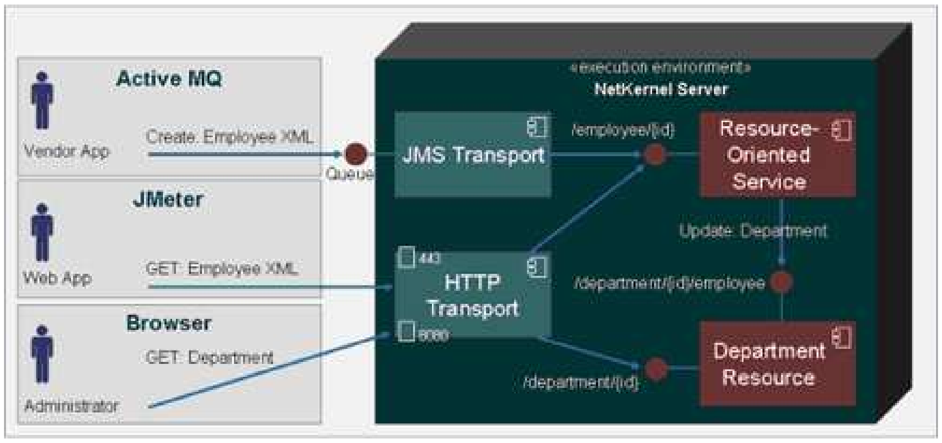


Figure 3 Resource-Oriented Service Demo

When a consumer requests a department resource, the returned representation contains links to the employee resources. The consumer can then extract the relative URI, /foo/business /employee/1234, to update that employee resource via JMS and, in turn, that action would result in a sub-request to update the department resource.

```
<?xml version="1.0" encoding="UTF-8"?>
<department>
<id>management</id>
<staff>
<employee>http://localhost:8060/foo/business/employee/1234</employee>
<employee>http://localhost:8060/foo/business/employee/4321</employee>
</staff>
</department>
```

Figure 4 Department Resource Representation

Implementing ROA provides an organization with *enterprise architectural agility* because it results in a logical layer of interconnected resources. This logical layer is decoupled from the physical implementation allowing consumer and provider to evolve independently over time. In addition, ROA pushes integration functionality to the edge of the network (as a URI), translating into better service management and scalability.

Jeremy Deane is a Technical Architect at Collaborative Consulting and has over 12 years of software engineering experience in leadership positions. His areas of expertise include Resource Oriented Architecture, Performance Engineering and Software Process Improvement. He can be reached at jdeane[at[collaborative.com .

[1] Gartner ID Number: G00161125

[2] 1060 Netkernel

[3] Apache Active MQ

[4] Collaborative Consulting download page

[5] Basic or Digest Authentication

[6] Spring Framework

[7] Ruby on Rails

[8] Introduction to Resource Oriented Computing

Back to Articles, including SOA, BPM & BA

READ MORE ON SOAINSTITUTE.ORG

Featured Presentation:

Case Study: Role of SOA in Legacy Transformation
Featuring: Todd Richmond, Vice President Enterprise Architecture, Sabre Holdings

Try changing out the engine of an airplane cruising at an altitude of 30,000 feet. Sabre Holdings is doing just that. As the world's largest global travel distribution system, Sabre supports...

more

Featured White Paper:

SOA and BPM - Taking the Enterprise to the Next Level
Courtesy of: Rick Sweeney, Independent Consultant

It is unfortunate but most companies find themselves always trying to catch up with technology advancements. Like many others they are burdened by a significant investment in legacy systems (and the...

more