If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

RESTful Services with Java

Concepts and Hands On

Mahesh K Punjabi

What is a Web Service

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

Evolution of Web Services

Web Services evolved from a long history of distributed programming

EDI
Electronic Data Interchange
1990

Computer to computer communication, with standard message format. Generally used for business transactions like invoices, purchase orders

RPC
Remote Procedure Call
1990

Call a procedure, function or a method on a remote computer, as if it exists locally.

COM-DCOM
Distributed Component Object Model
1990

Based on RPC, added features like object inheritance. COM and DCOM with OLE were Microsoft technologies for distributed computing

CORBA
Common Object Request Broker Architecture
1991

A CORBA program written in any language on one computer can interoperate with any other CORBA program written in any other language. Very Complex!

Java RMI
Remote Method Invocation
1996

One JVM communicating over IIOP – Internet inter orb protocol to invoke a method running in another JVM

HTTP
Hypertext Transfer Protocol
1996

Plain text, easy to read, formatted text organized by tags. This was precursor to SOAP, created by same team (at Microsoft) which created SOAP

XML-RPC
Simple Object Access Protocol
1998

Plain text, easy to read, formatted text organized by tags. This was precursor to SOAP, created by same team (at Microsoft) which created SOAP

SOAP
Simple Object Access Protocol
2000

Evolved from XML-RPC. Became foundation of what is known as 'Web Services'. There were several WS-* standards around this. SOAP communicates on top of HTTP

REST
Representational State Transfer
2000

Became more popular recently, for invoking remote services directly over HTTP

HATEOS
Hypermedia as the engine of Application State
2000

Part of Roy Fielding thesis. Principle is that the client interacts with a network application entirely through hypermedia provided by server. This is opposed to principle of client and server interacting via a fixed interface shared through an interface definition language (like WSDL)

ODATA
Open Data Protocol
2007

An open protocol which allows creation and consumption of queryable and interoperable RESTful APIs. Shares similarities to JDBC and ODBC

OAS
OpenAPI Specification
2010

Originally known as Swagger Specification, it is a machine-readable interface for describing, producing, consuming and visualizing REST web services.

So what is a Web Service?

Web Service = Web (HTTP) + Service (SOAP/REST/XML/JSON/Text)

A RESTful web service is a service that communicates over web (HTTP) and adheres to the REST Architectural Constraints

The Origin of REST

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

- REST stands for Representational (RE) State (S) Transfer (T)
- It is a software Architectural Style for developing Distributed Systems
- It came out first in the PhD thesis of Roy Fielding, at University of California, Irvine

Who is Roy Fielding?

URL: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

Roy Thomas Fielding

Education

Doctor of Philosophy (2000)
University of California, Irvine
Information and Computer Science
Institute of Software Research
Advisor: Dr. Richard N. Taylor
Dissertation: *Architectural Styles and the Design of Network-based Software Architectures*

Master of Science (1993)
University of California, Irvine
Information and Computer Science
Major Emphasis: Software

Bachelor of Science (1988)
University of California, Irvine
Information and Computer Science

Professional Experience

12/99 -	Chief Scientist, eBuilt, Inc., Irvine, California
3/99 -	Chairman, The Apache Software Foundation
4/92 - 12/99	Graduate Student Researcher, Institute for Software Research University of California, Irvine
6/95 - 9/95	Visiting Scholar, World Wide Web Consortium (W3C) MIT Laboratory of Computer Science, Cambridge, Massachusetts
9/91 - 3/92	Teaching Assistant ICS 121 - Introduction to Software Engineering ICS 125A - Project in Software Engineering University of California, Irvine
11/89 - 6/91	Software Engineer ADC Kentrox, Inc., Portland, Oregon

The REST Architectural Style – What It Is

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

- REST is an Architectural Style, governed by certain constraints to build Distributed Applications

Constraints Requested by REST → *The real reason why REST became prominent*

- Loosely Coupled Client & Server : Both client and server will evolve separately, connected by a common interface
- Stateless : All services should be stateless. Thus each request should be independent of each other, and not build up on assumption of having some data in http session. → *No more HTTP Session!*
- Cacheable: The networking infrastructure should be able to support cache at various levels
- Uniform Interface: All the interactions between client and server should follow a generic interface. Each resource exposed by the server should have a unique address and should be accessible by a generic interface. The client should be able to act on resources by using generic set of methods (***HTTP Methods or Verbs***).

This constraint is the center piece of REST

- Layered System: The server can have multiple layers of implementation, like load balancers, caches, security policy implementations etc.
- Code On Demand: This constrain is optional. It means that client applications can download code from server to be executed at runtime on the client side. → *Made for JavaScript frameworks…*

The REST Architectural Style – What It is Not

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

- REST does not dictate what kind of technology to use [*It only defines how data is to be transferred between client and server*].
- REST does not dictate a networking protocol [*But instead of implementing new protocols it simply uses WWW to create RESTful architectures, we call it REST over HTTP*]
- Even before all REST constraints were finalized, there were several working REST applications [*Example: All static websites are RESTful*]

How are static websites RESTful

- All the links in a static website points to a resource like a Page, Image, Video etc.
- Static websites can use caching on browser and server [Remember No-Cache meta tags]
- A static website connection to server is stateless

So what is the contribution of Roy Fielding in REST?

- Roy Fielding put the constraints for using REST for Dynamic Web Services for Creating Distributed Systems
- Using the available system (WWW), for light weight distributed programming, without overheads of heavy protocols like SOAP and making services as simple as static resources

REST Terminology

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

Uniform Interface constraint is the key feature that distinguished REST applications from other network-based applications. Uniform interface in a REST application is achieved through the following Abstractions

Resources

Fundamental to REST is the concept of resource.

- A resource is anything that can be accessed or manipulated
- Example of Resource is : Image, Video, Blog entries, User profiles, a person, a device etc. (I think : we can also call a resource an entity)
- A resource can be related to corresponding other resources, for example a customer can place order, so customer and order are related.
- Resources can be grouped in a collection, for example orders is a collection of individual 'order' resources

URIs (For Identifying Resources)

Before we can interact with a resource we should be able to identify it

- The web provides concept of URI – Uniform Resource Identifier for uniquely identifying resources. A URI is formatted as scheme:scheme-specific-part
- In http://www.nytimes.com/2342, here http is the scheme. It indicates that http scheme should be used for interpreting the rest of the URI.

The following table gives examples of URI and how resources are identified

URI	Resource Description
http://blog.example.com/posts	A collection of blog posts
http://blog.example.com/posts/1	A blog post identified by 1
http://blog.example.com/posts/1/comments	All comments for a blog post '1'
http://blog.example.com/posts/1/comments/3	Comment identified by '3' of blog post '1'

URI Templates : The template for above is http://blog.example.com/posts/{postId}/comments/{commentId}

REST Terminology

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

Representation

- RESTful resources are abstract entities
- The data and metadata that makes a RESTful resource, needs to be serialized into representation before it is sent to a client.
- This representation can be considered as a snapshot of the resource's state at a given point of time.
- For example the actual data is in a database. What the client gets is XML or JSON data. This XML or JSON data is the representation of the actual data.
- REST components (clients) interact with a resource by transferring its representation back and forth. They never interact directly with the resource [Philosophical point]
- The same resource can have several representations, like HTML, JSON, XML or binary formats such as PDF, JPEG, MP4 etc.

HTTP Methods

- The 'Uniform Interface' constraint restricts the interactions between client and server through a handful of standardized operations or verbs.
- On the web HTTP standard provides eight HTTP methods that allow the clients to interact and manipulate the resources.
- Note: HTTP verb and HTTP methods are synonymous (they represent the same concept/thing)

Attributes of Consideration for HTTP Methods

- Safety: A HTTP method is considered to be safe, if it does not change the state of the resource (don't consider logging as changing the state). GET or HEAD are safe methods (it depends on the implementation however, after all)
- Idempotent: An operation is considered idempotent if it produces same server state whether we apply it once or multiple times. So GET is idempotent, so is HEAD. Because POST will insert for example an order, by executing same POST method the state will keep on changing. Thus POST is not considered idempotent.

Understanding HTTP Request Methods

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

GET Method

GET method is safe and idempotent. Because it is so, the response to GET request can be cached.
People may use simplicity of GET method to abuse and perform operations like deleting or updating a resource representation. Such usage violates the standard HTTP semantics and is highly discouraged

HEAD Method

There are times when you want to check if a particular resource exists but we don't really care of the representation.
Or there are times when you want to know if a new version of the resource is available or not, before you actually download its representation.
In both the above cases GET will be considered as heavy weight. Instead we use HEAD

DELETE Method

This method is used for resource to be deleted (or marked as deleted).

PUT Method

Put method is used to modify (update) a resource.

POST Method

Post method is used to create a resource.
Typically post is used to add to a collection or sub-collection of resources.

CRUD and HTTP Verbs

1. Create -> POST
2. Update -> PUT
3. Read - > GET
4. Delete -> DELETE
5. Use your discretion however with POST and PUT, as you can use POST for modifications as well

Understanding HTTP

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

HTTP stands for Hypertext Transfer Protocol. It is the foundation of WWW. This protocols defines how messages are

(a) formatted (b) transmitted (c) processed over the internet.

HTTP Versions

- HTTP/0.9 : Released in 1991. It only supported GET method
- HTTP/1.0 : Released in 1996. It supported GET, HEAD, POST and other method
- HTTP/1.1 : Released in 1999.
- HTTP/2 : This version was released in 2015. Most browsers have added support for this now.

HTTP's Request/Response

Demo: *SOAP UI for a simple request/response*

URI vs. URL

- **URI :** Uniform Resource Identifier : It is a 'text' that identifies any resource or a name on internet
- **URL:** A URI can be a URL, if the 'text' used to identify the resource also includes protocol for accessing it like HTTP or FTP

All URLs are URIs

Java Frameworks for Building REST Web Service

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

The following frameworks are available for creating REST web services

- **JAX-RS :** Java API For RESTful web services (Part of Java EE). This specification has several implementations as
  - Jersey REST
  - Apache CXF
  - RESTEasy (from JBoss)
  - Restlet

JAX-RS was first introduced in Java EE 6. JAX-RS 2 was introduced in Java EE 7 (support for HATEOS, Validations, Filters)

Apart from JAX-RS based frameworks, there are some non-standard Java REST Frameworks (***which do not abide by JAX-RS specification***)

- RESTX : Comparatively new entrant in the market
- Spark :
- Play : Java and Scala based Web application framework, which supports building RESTful web services

Message Formats

- REST does not prescribe any specific message format for client-server communication
- One can use an appropriate format for representing messages as long as it is supported by HTTP
- XML and JSON are two most popular formats used by RESTful web services today. Out of these JSON is lot more popular and widely adopted by enterprises because it is simple, lightweight.

Thought : Why Spring MVC is not mentioned here?

Refer: Using JAX-RS with Spring Boot

JSON Overview

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

- JSON is lightweight, text based format. It stands for Javascript Object Notation, but it is not bound to Javascript as such. It can be used by any language because most languages support JSON.
- JSON is a collection of **name, value pairs**. Name and value are separated by colon (:). Name is string and values can be any JSON data type like number, string, Boolean, array or null. Multiple pairs of name and value are separated from each other using a comma. The entire collection of name value pairs is enclosed in curly braces {} and represent an object.
- A JSON value can also be an array. In such cases the values are enclosed by [] and separated from each other by comma (,)

```
{
"emp-no": 98764,
"first-name": "Ramesh",
"last-name": "Rao",
"unit-name": "FS",
"join-date": "1-Jan-2015",
"years-of-experience": 12,
"location": "Delhi"
}
```

JSON Data Types

- String
- Number
- Boolean
- Array
- Object
- null

Java API for JSON Processing

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

There are many Java based frameworks for processing JSON. Here we will discuss the API available in Java EE platform for processing JSON.

Java EE 7 has standardized JSON processing APIs with **JSR 353**- Java API for JSON Processing

JSON-P	*Java API for JSON Processing. P stands for processing (different from JSON-P : Padding). Erstwhile default engine for JSON processing*
MOXy	*JSON binding support via MOXy is a default and preferred way of supporting JSON binding in your Jersey applications since Jersey 2.0*
Jackson	*Jackson is a multipurpose library for processing JSON. It also has support for multiple other formats.*
Gson	*Gson is open source Java library from google, for converting Java objects into JSON and vice versa*

Creating a RESTful Java Service : Steps

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

The following are the various logical steps to create a RESTful service, using Eclipse and Tomcat

1	Environment Setup	**Create a Dynamic Web Project in Eclipse**	**Configure it to support Maven**	
2	Enabling REST	**Modify web.xml to enable REST**	**Modify web.xml to enable default JAXB provider**	
3	Writing Service and Model	**Create Java Service, Add VERB methods, Add Annotations**	**Create Model for your Service. Add JAXB annotations**	**Configure and Write DataStore /JPA Code**
4	Enhancements to Service	**Enable Logging Using Log4j2 (the new version)**	**Write Adapter Classes for Data Conversion be Java & JSON**	
5	Test Service	**Test Using POSTMAN, SOAPUI, FireBug or any other REST Client**		

JAX-RS Annotations

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

- @Path : javax.ws.rs.Path indicates the URI path to which the class or method will respond. This path is relative to the path of server. The annotation can be defined at class and method levels. We can also specify a regular expression by using syntax like @Path("{name: [a-zA-Z][a-zA-Z_0-9]}")

- @Produces: javax.ws.rs.Produces annotation is used to define the media type that the REST method will return. Example of such a media type can be : application/json or application/xml or text/plain

- @Consumes: javax.ws.rs.Consumes annotation is used to define the media type that the method will accept.

- @GET: To designate the HTTP verb GET corresponding to a method

- @OPTIONS: This annotation designates a method to return a list of HTTP methods allowed on a resource. JAX-RS offers the default implementation for the OPTIONS method type

Annotations for Accessing Request Paremeters

Path Parameter /employee/{id} @PathParam(“id”)

Query Parameter /employee?name=Ramesh&empNo=1232 @QueryParam(“id”)

WADL & Swagger

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

Jersey contains the support for Web Application Description Language (WADL). It is an XML description of deployed RESTful web application. In a way WADL is similar to WSDL, except that it describes a restful service

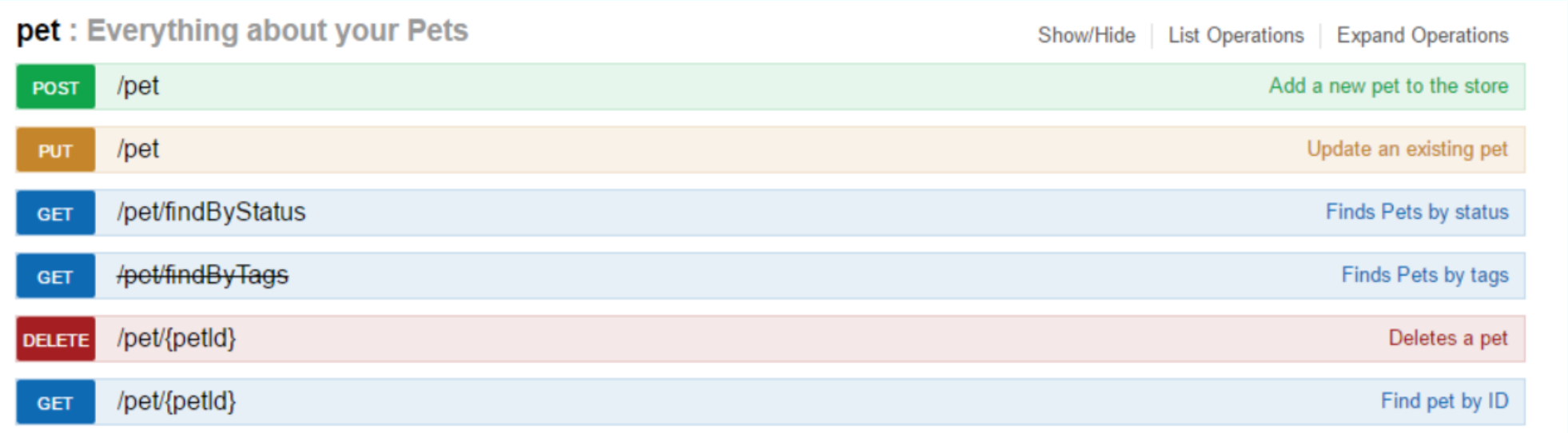
Try the following with Demo Example

http://localhost:8080/cms/rest/application.wadl

http://localhost:8080/cms/rest/application.wadl?detail

Swagger

Easy way to document your web services



How Swagger Works

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

Top Down Approach

Use Swagger Editor to create your **Swagger Definition** and use the Integrated **Swagger Codegen** tools to generate server implementation

Bottom Up Approach

You can create definition manually using the **Swagger Editor** for REST services which are not JAX-RS compliant

For JAX-RS compliant web services or Node.JS web services you can get the Swagger definition generated automatically for you.

For Web Service Consumers

If you have an API that has Swagger definition, you can use online version of **Swagger UI** to explore the API and then use Swagger Codegen to generate the client library of your choice

Swagger Offerings

Look at the several open source [http://swagger.io/getting-started/open-source-integrations] and commercial [http://swagger.io/getting-started/commercial-tools] offerings available for Swagger.

Security in RESTful Web Services

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

HTTP Basic Authentication : Here we send a Base64 encoded Username and password (over SSL) as an HTTP header. This must be sent for every request

Digest Authentication : Instead of password, the client sends a cryptographic hash of user credentials along with user name.

Client Cert Authentication : Here we use a security certificate or other custom token to authenticate a user

OAuth: Open standard for authorization. Here the concept is to have Google or Yahoo (a third party service provider) provide the authorization token.

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

Why REST

- Simpler than SOAP
- No extra frameworks/libraries needed on the client side
- Permits multiple data formats
- Much lightweight, better for performance and scalability
- Has replaced SOAP across all the big enterprises

Why SOAP

- WS-Security : Beyond SSL, WS security adds identity passing through intermediaries
- WS-AtomicTransactions : Transaction handling over a service, use SOAP
- WS-ReliableMessaging : Ensures that message is delivered. In REST you have to take care
- If you have legacy service, which you have to invoke from server side

REST Best Practices for API Design & Coding

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

- Use Noun and Not Verbs in the URL:
- The Endpoint name should be plural:
- Always use SSL for transport channel security:
- Document Your Web Services
- Make them oData Compliant
- Versioning of Web Services
- Use Query Param for Filtering, Sorting and Searching
- Integrate with oAuth
- Include versioning in the URL
- Use Interfaces for your API methods
- Use Error Payloads for exception, including an error code and the message sent together as JSON object

If you are not able to join audio, its because of the limit on maximum number of participants. Do not **despair!**. The deck, code zip and the video recording of this session will be made available at http://star/rest

The important thing is to not stop Questioning

- Einstein

Questions