

Service Oriented Programming

Javier Espinosa, PhD

javier.espinosa@imag.fr

Reminder

<i>Communicating entities (what is communicating)</i>		<i>Communication paradigms (how they communicate)</i>		
<i>System-oriented entities</i>	<i>Problem-oriented entities</i>	<i>Interprocess communication</i>	<i>Remote invocation</i>	<i>Indirect communication</i>
Nodes	Objects	Message passing	Request-reply	Group communication
Processes	Components	Sockets	RPC	Publish-subscribe
	Web services	Multicast	RMI	Message queues Tuple spaces DSM

Outline

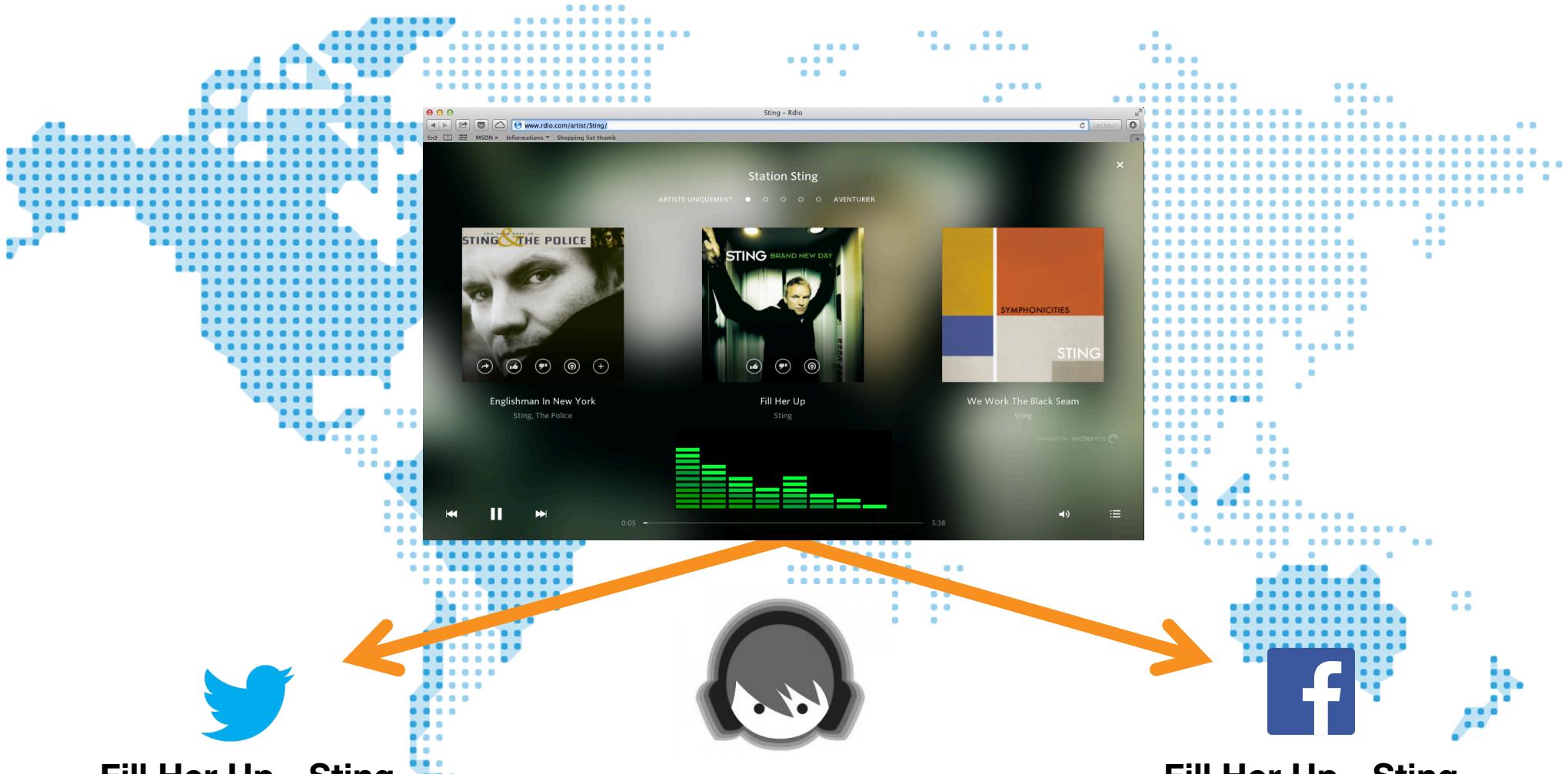
■ Service Oriented Computing

- Service
- Services Coordination
- Service Taxonomy

■ Service Implementation

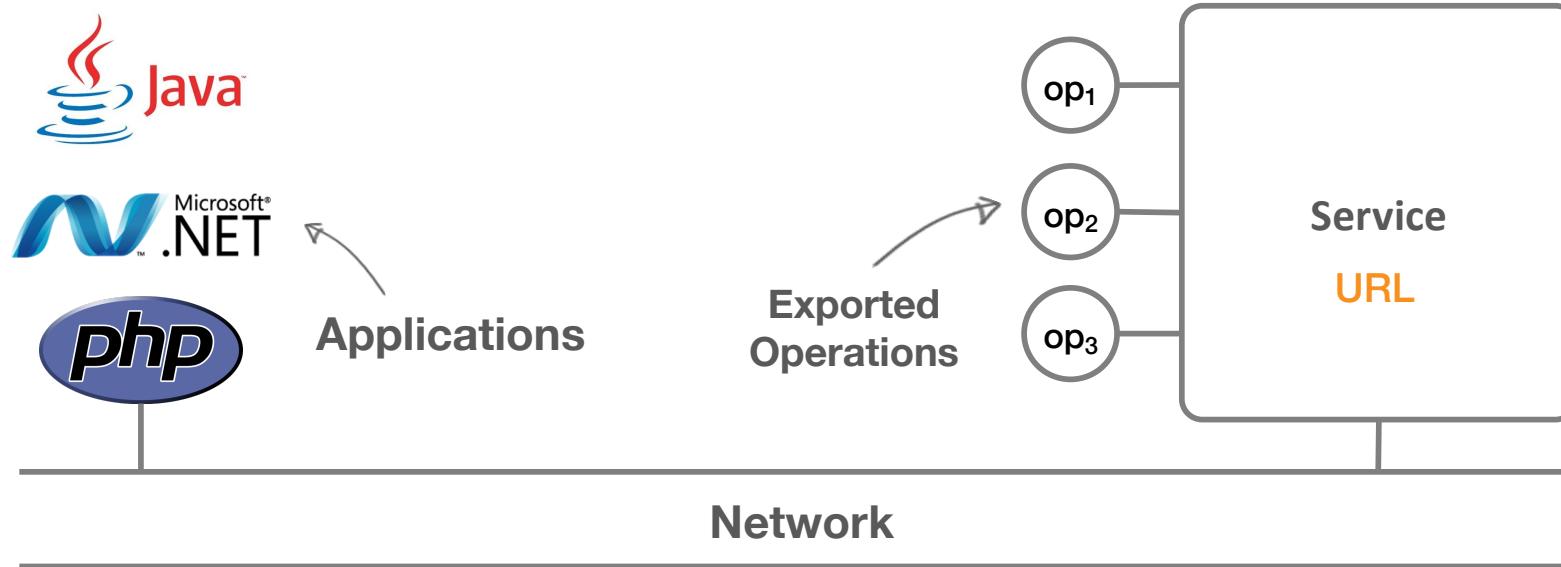
- Web Service
- RESTful Service
- Case study: CouchDB + Deezer





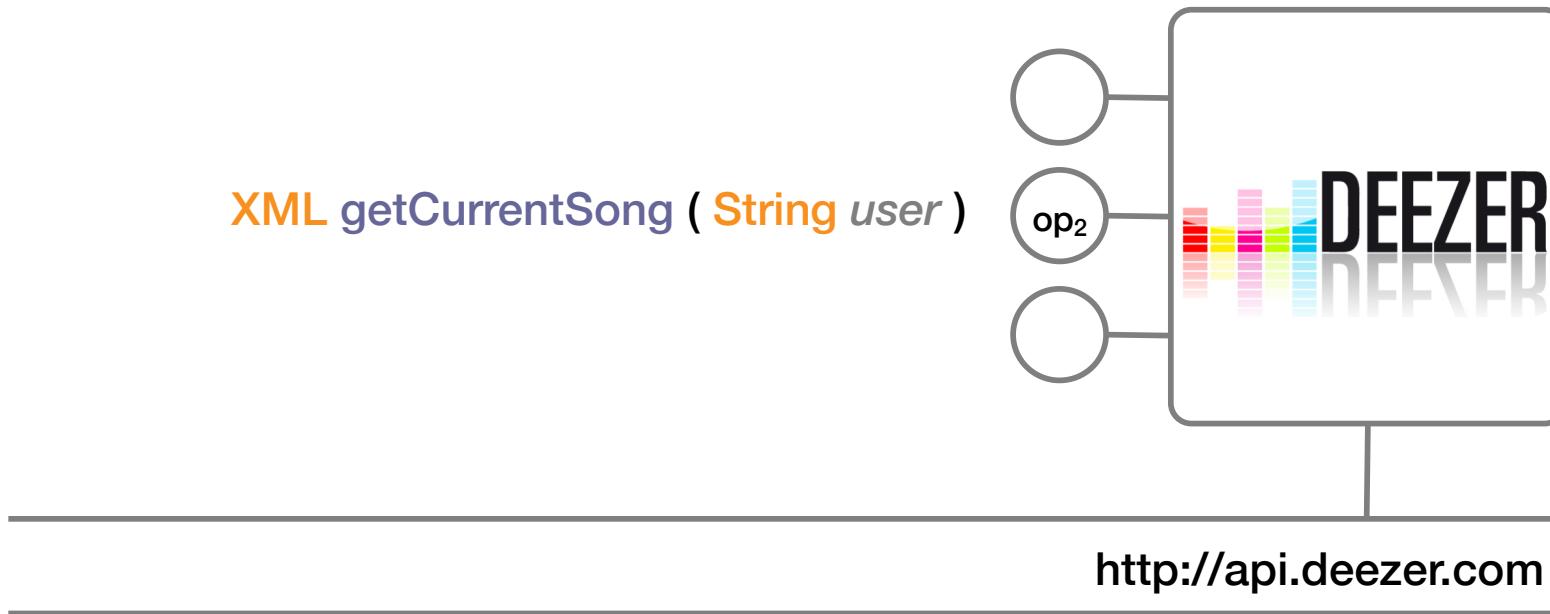
Service

- Component exporting operations accessible via a network

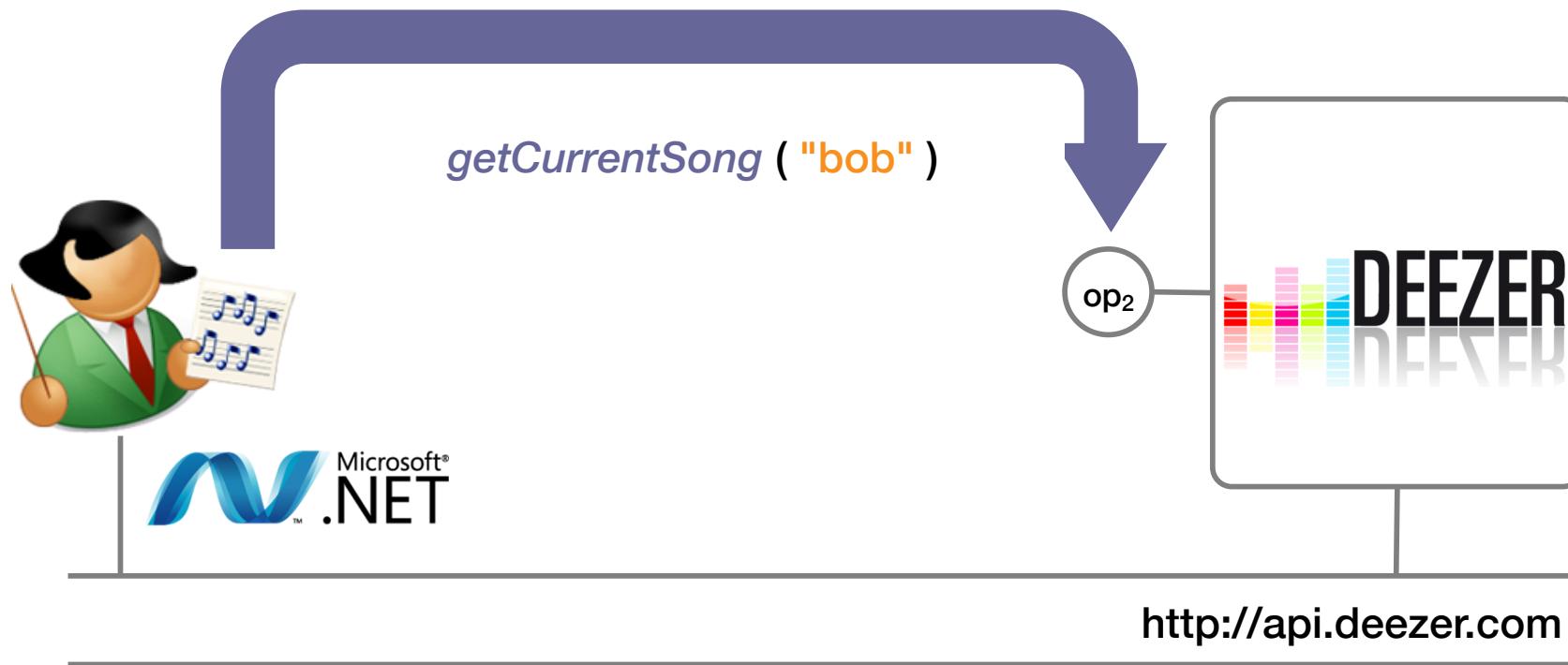


Service Example

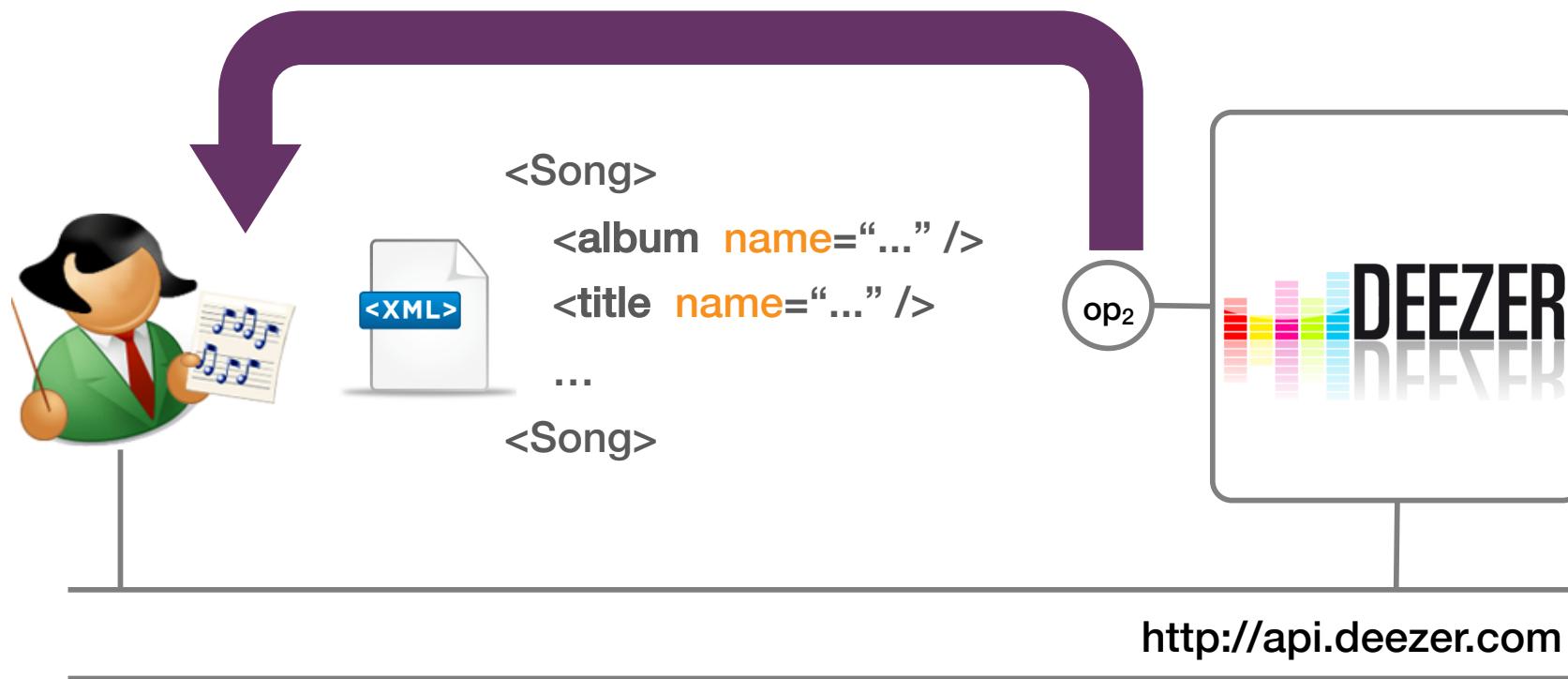
- " Retrieve the **current song** listened by a **Deezer user** "



Operation Call Example

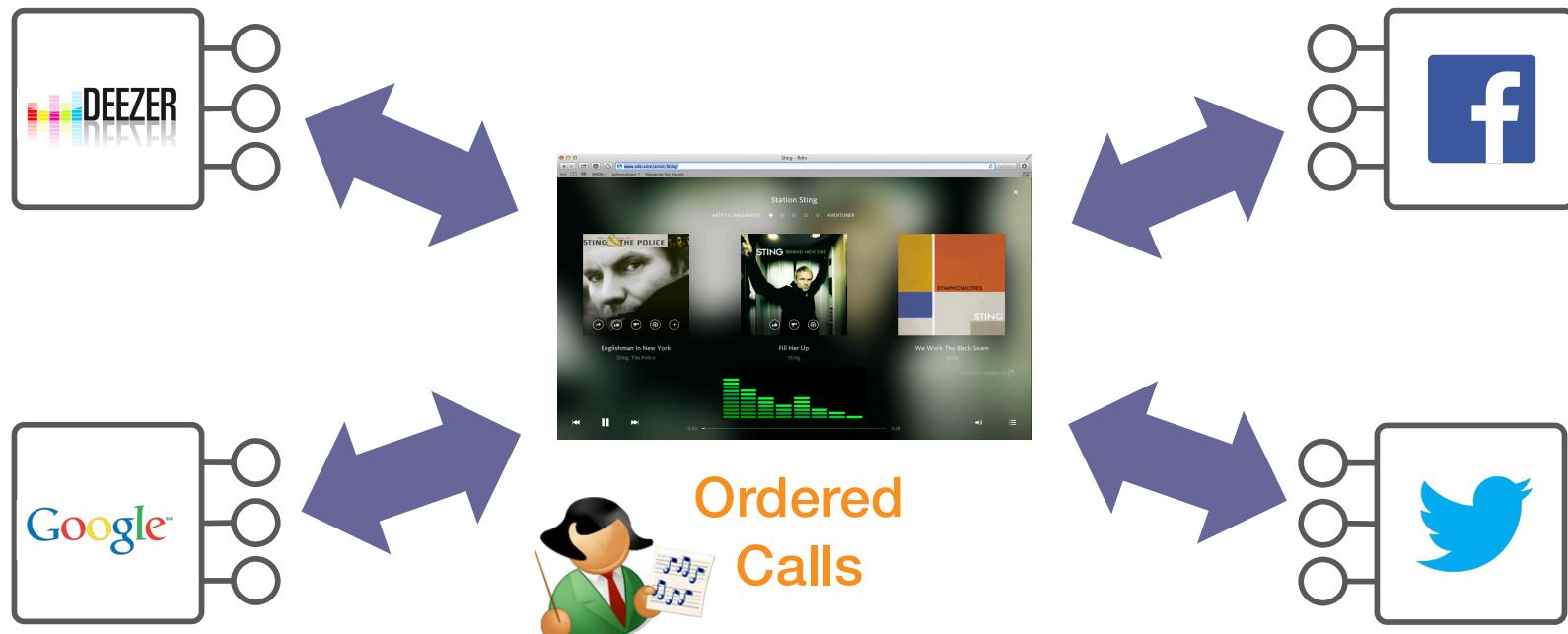


Operation Call Example



Services' Coordination

- Application synchronizing **ordered calls** to **services' operations**



Services' Coordination Implementation

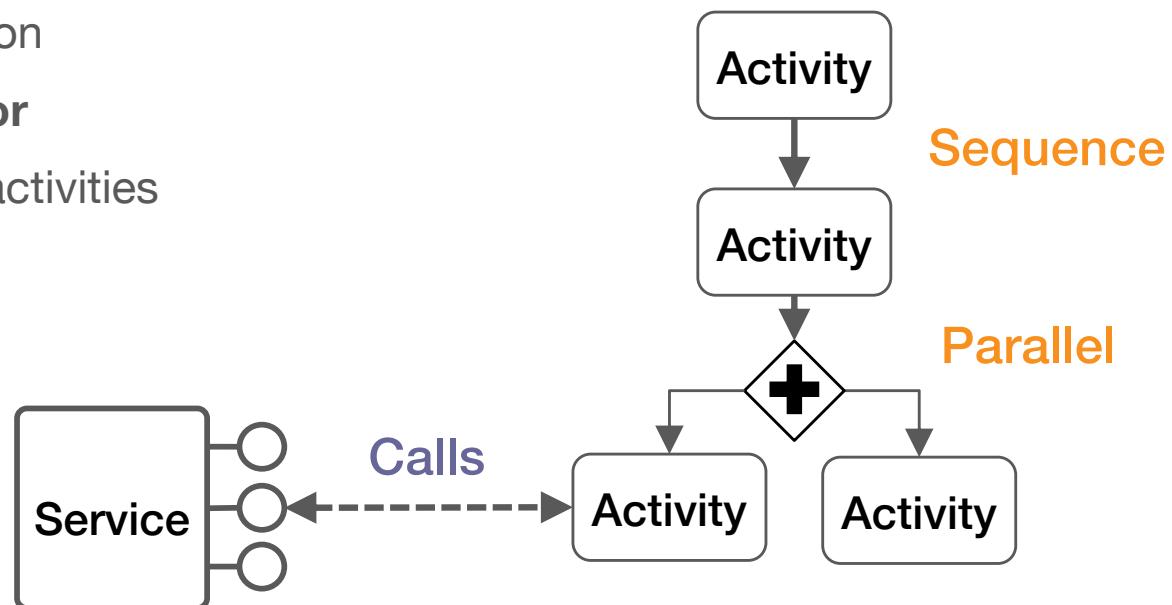
- Implemented as a Workflow (WF)

- **Activity**

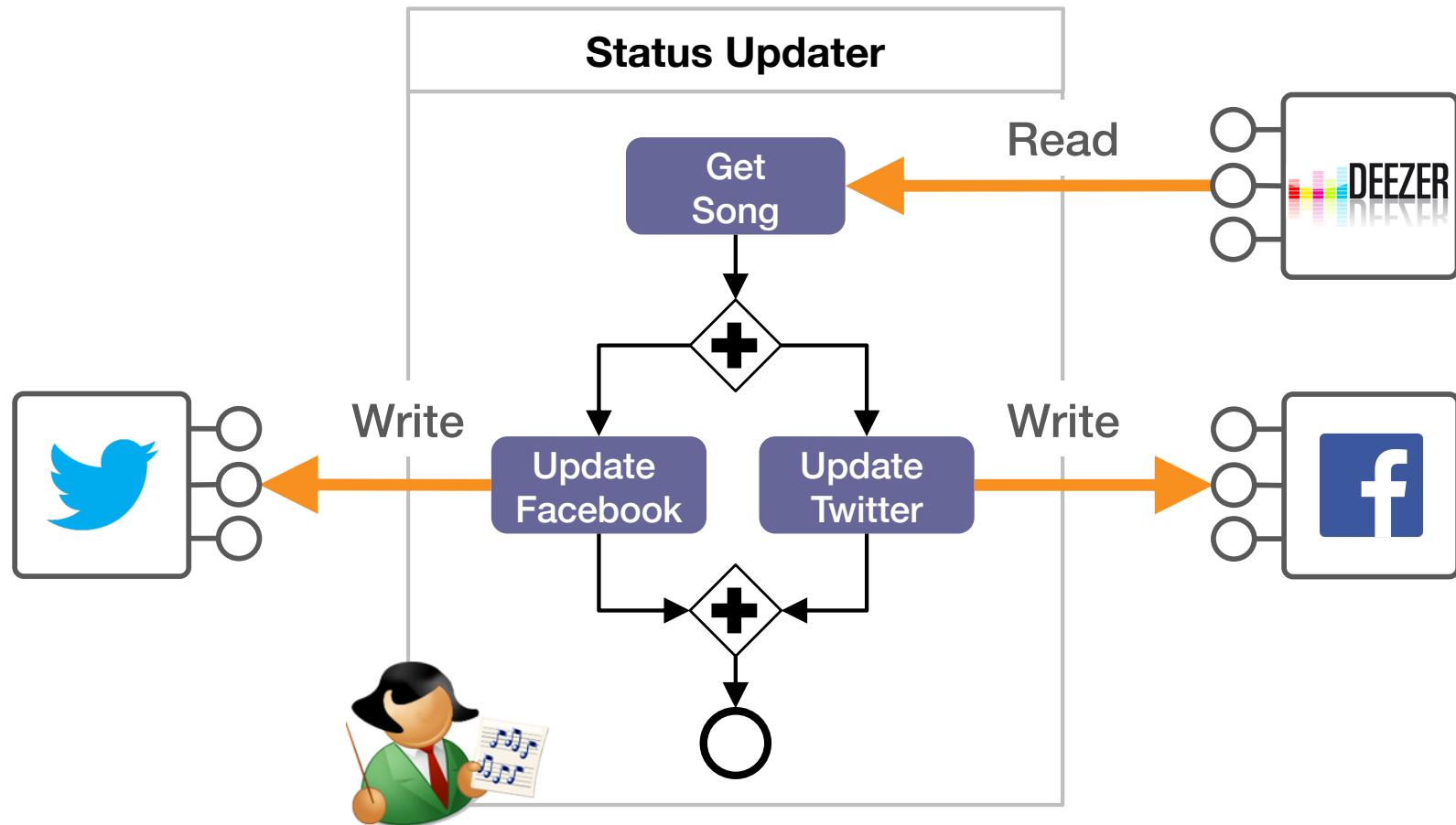
- Calls a service operation

- **Control Flow Operator**

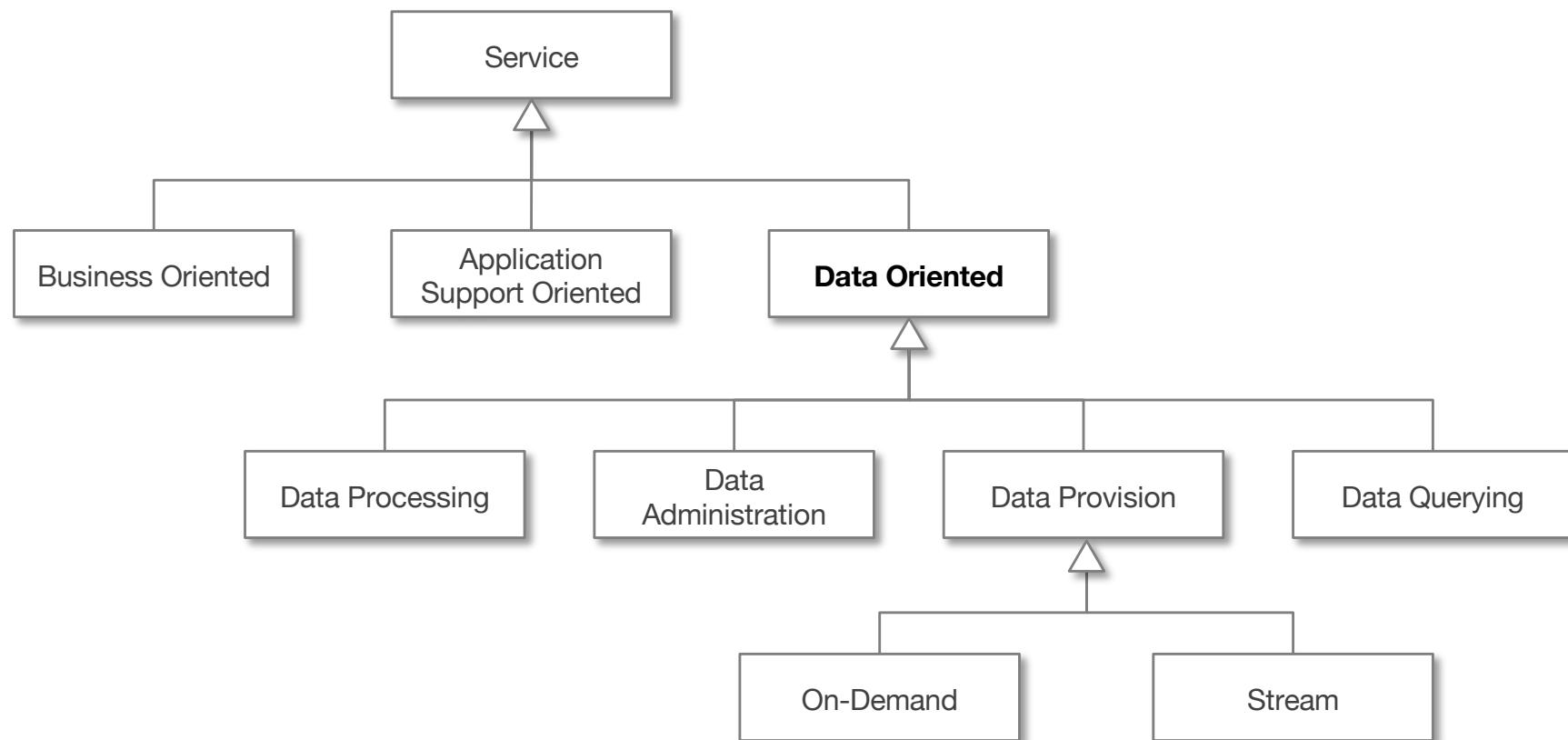
- Specify order among activities



Services' Coordination Example



Services Taxonomy (Overview)



Outline

- ✓ **Service Oriented Computing**

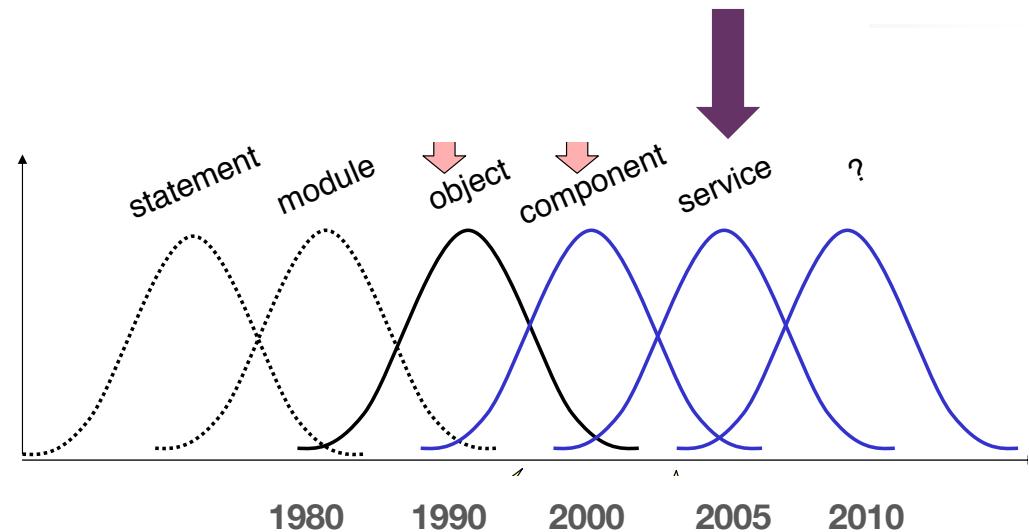
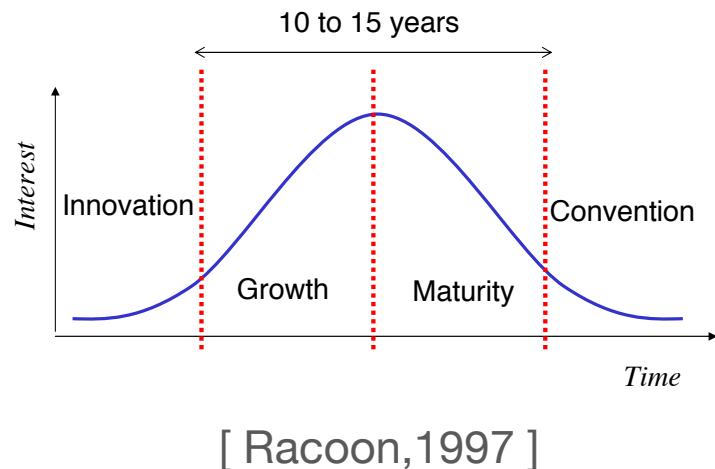
- ✓ Service
- ✓ Services Coordination
- ✓ Service Taxonomy

- **Service Implementation**

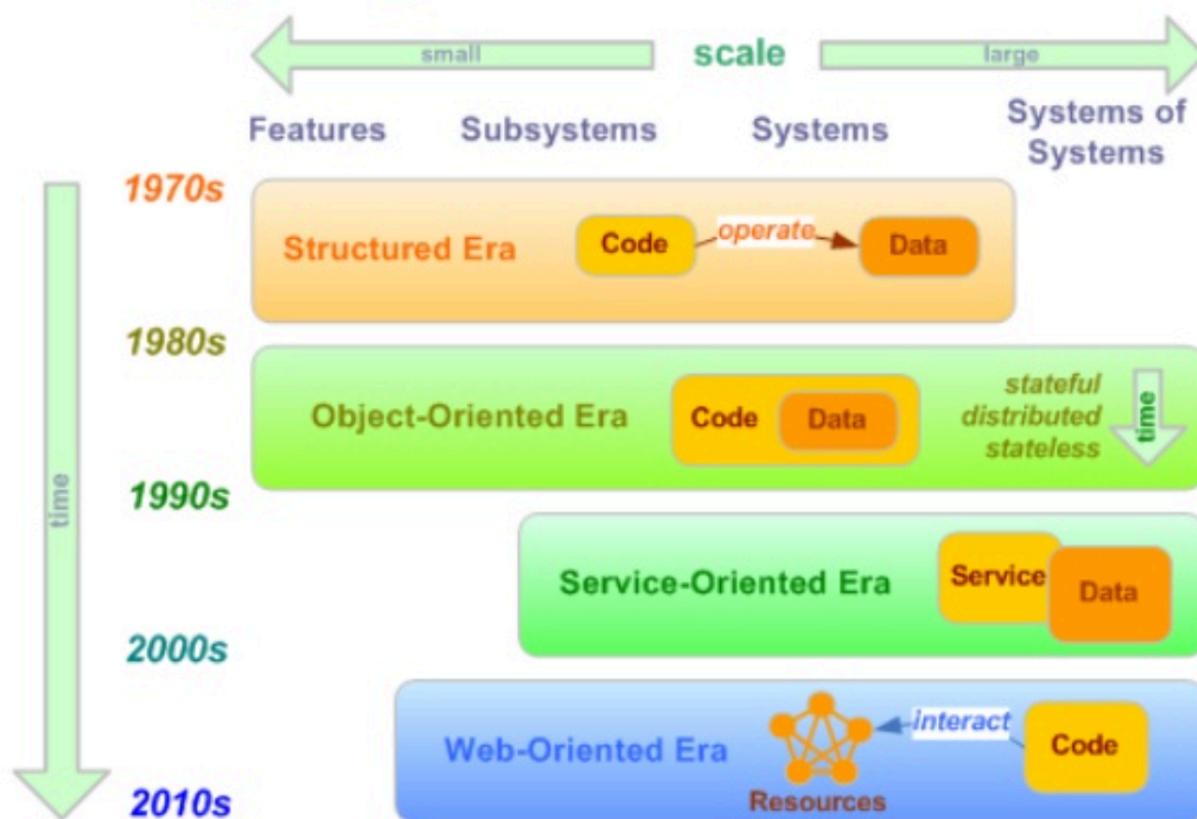
- Web Service
- RESTful Service
- Case study: CouchDB + Deezer

History

■ Evolution of Software Paradigms

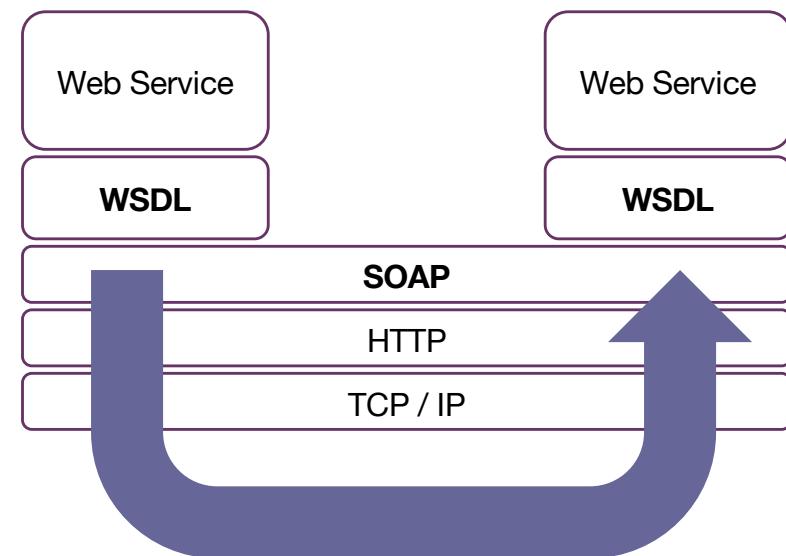


Popular Models for Developing and Integrating Software - 1970s to Now

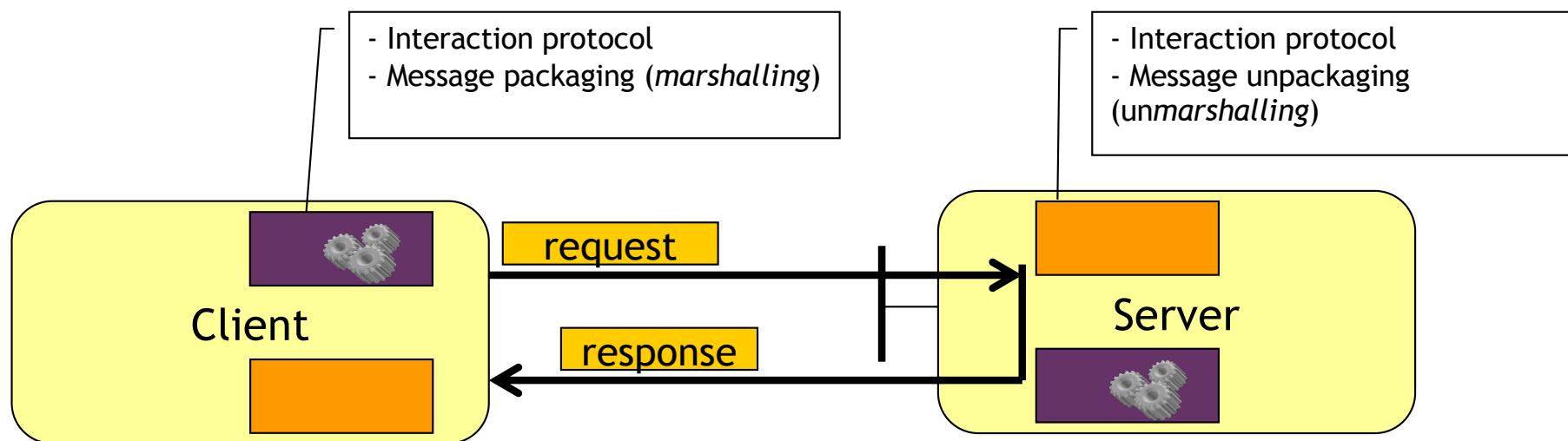


Web Service

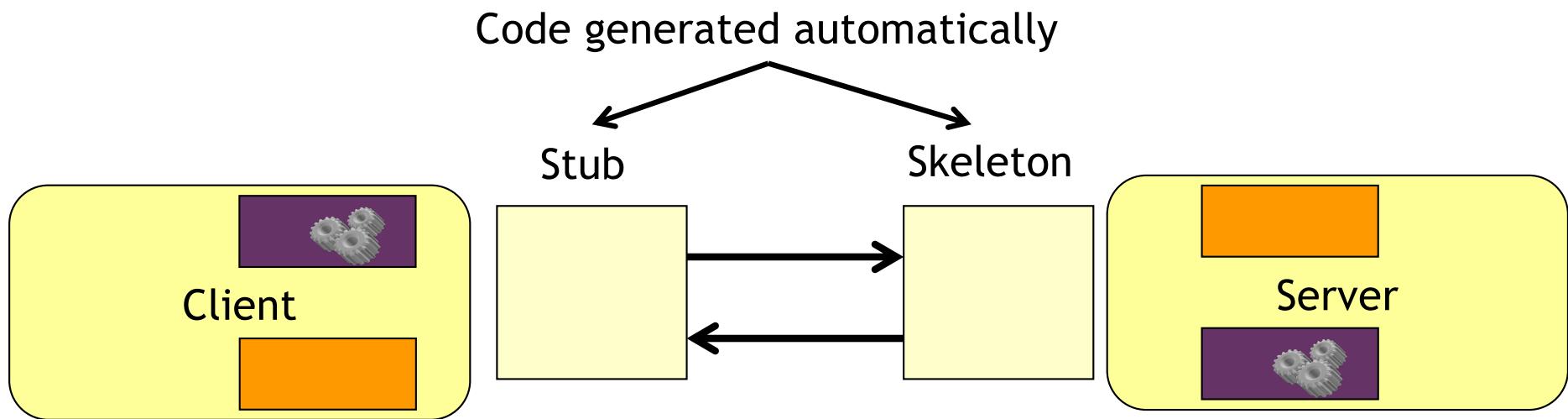
- Provides a **standard** means of **interoperating** between applications running on **different platforms**
- Exposes an **API** described in a **machine-processable format** (**WSDL**)
- Other systems interact with it using **SOAP** messages that are conveyed using **HTTP** and other **web-based technologies**



C/S: Low Level Primitives

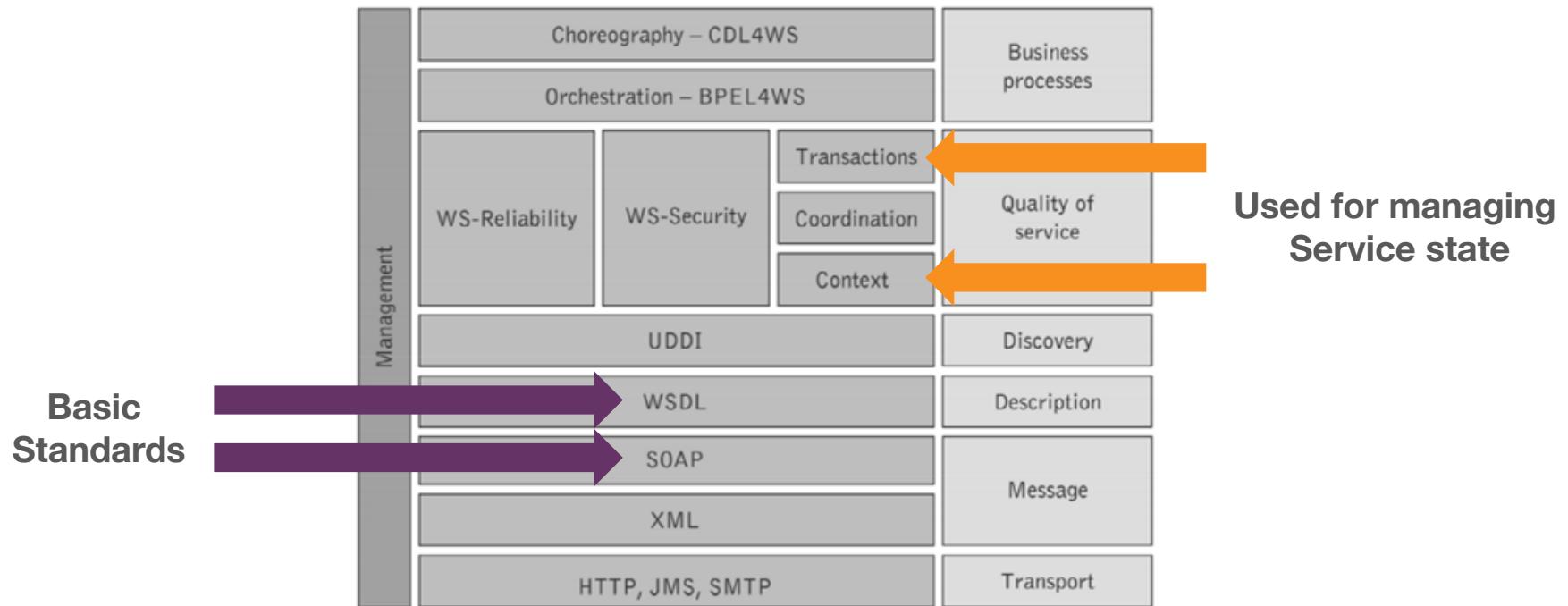


C/S: Middleware RPC



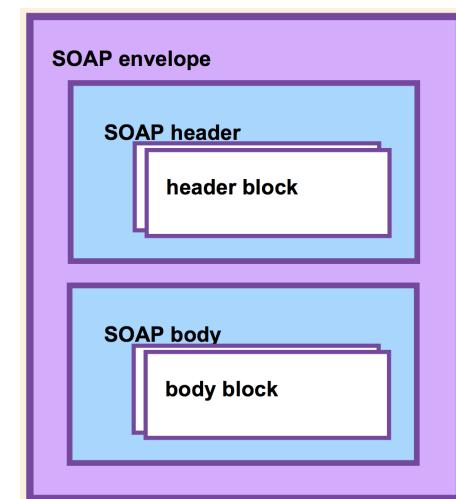
Web Service Protocol Stack

- Set of **standards** addressing interoperability aspects



Simple Object Access Protocol (SOAP)

- **Messaging Framework** for passing data back and forth between disparate systems
- Message structure:
 - **Envelop** defines what is in the message and how to process it
 - **Header** defines a set of encoding rules for expressing instances of application-defined data types
 - **Body** defines conventions for representing RMI calls and responses



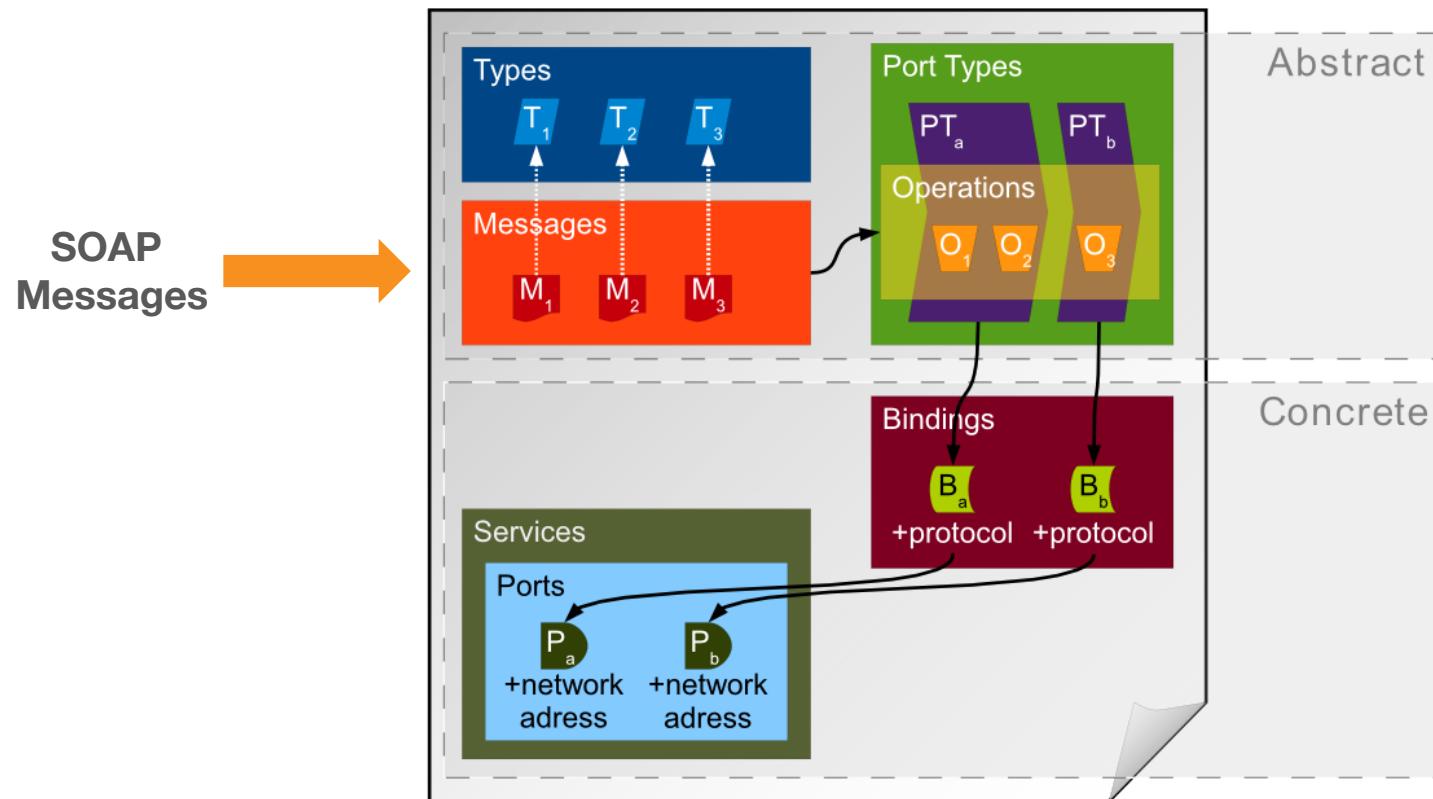
SOAP Message Example



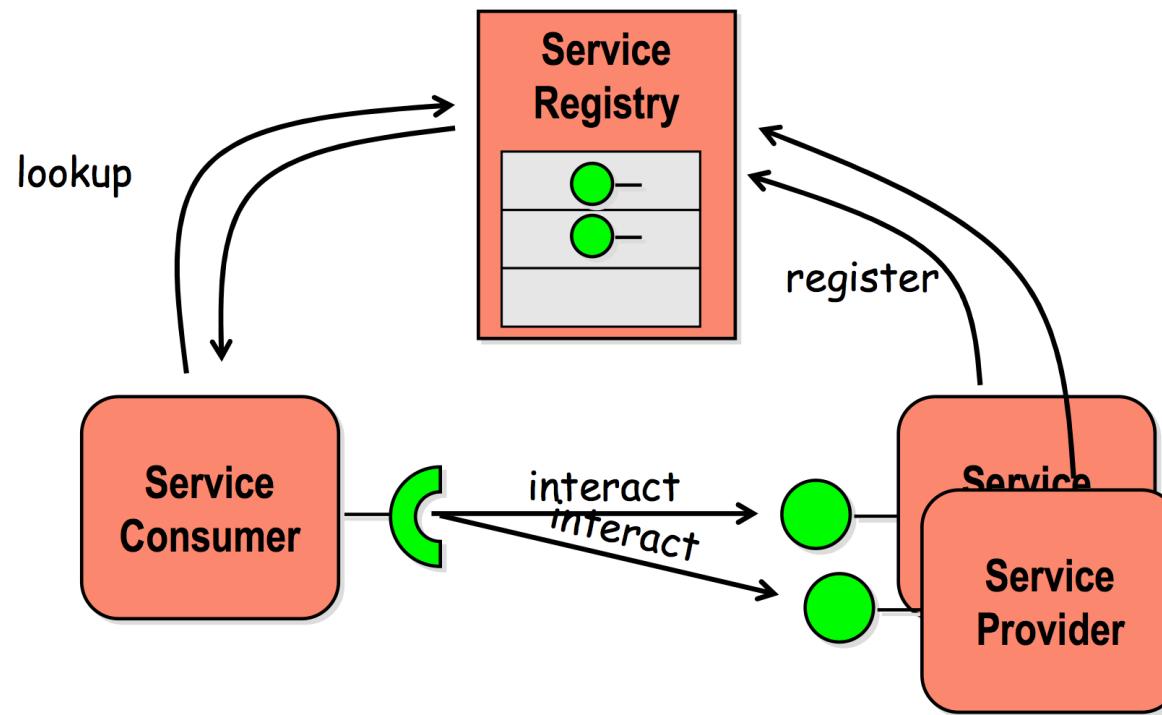
Web Service Description Language (WSDL)

- XML document that describes the mechanics for interacting with a particular service
 - **what** a service does
 - i.e., the operations the service provides
 - **where** it resides
 - i.e., details of the protocol and the specific URL
 - **how** to invoke it
 - i.e., details of the data formats and protocols necessary to access the service's operations

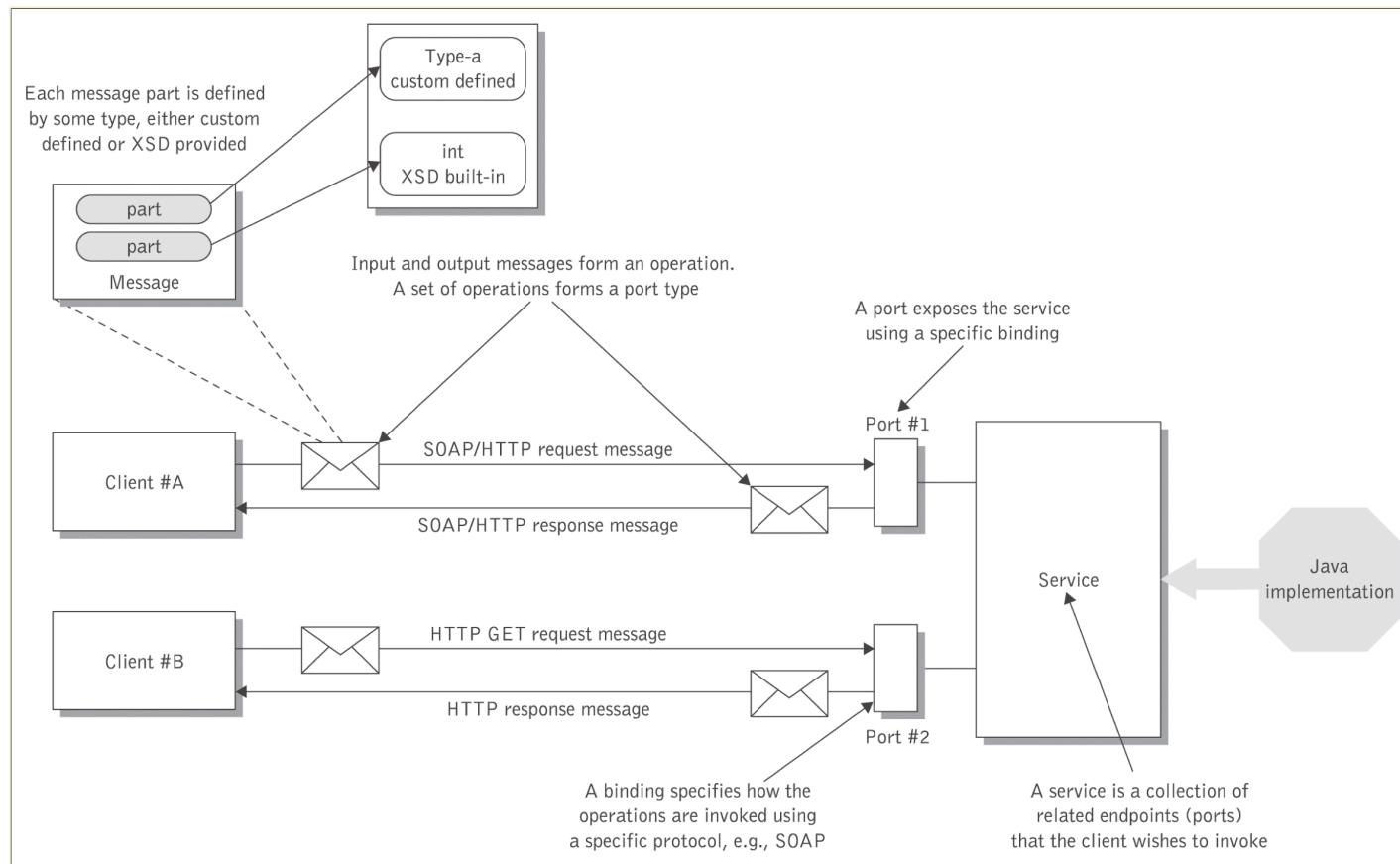
WSDL Structure



Web Service Architecture

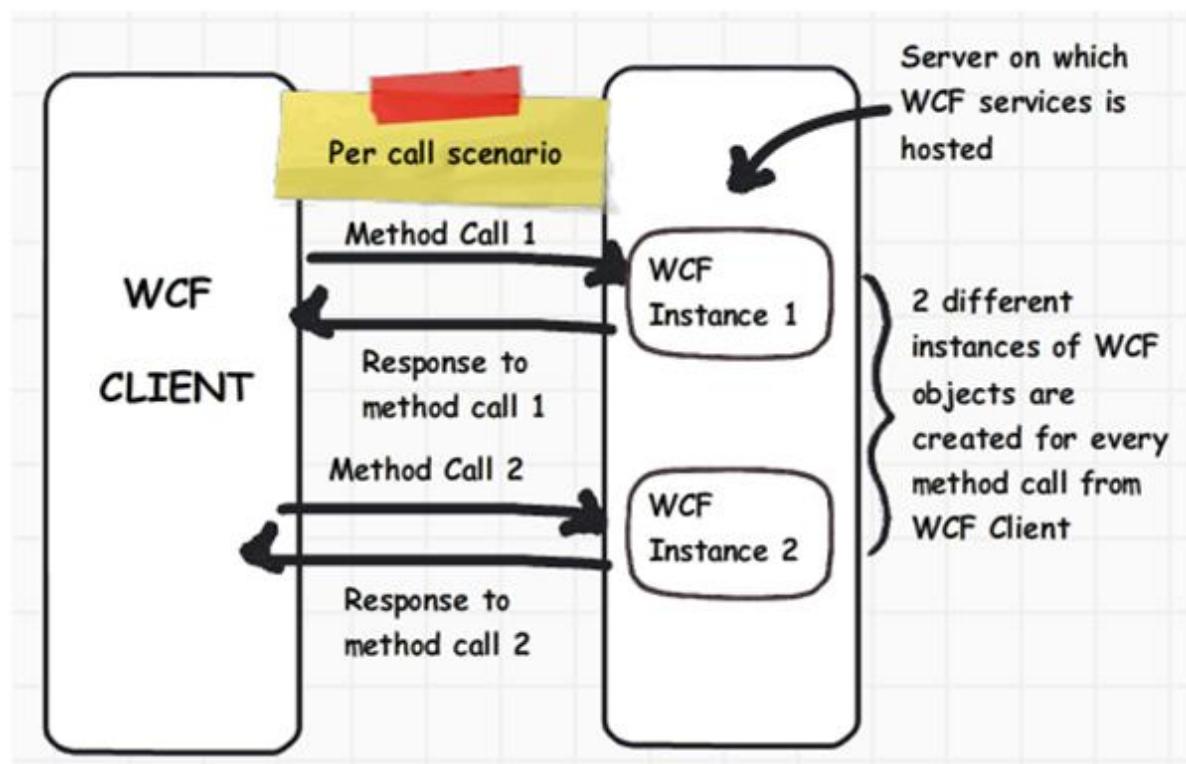


Example of Web Services Interaction

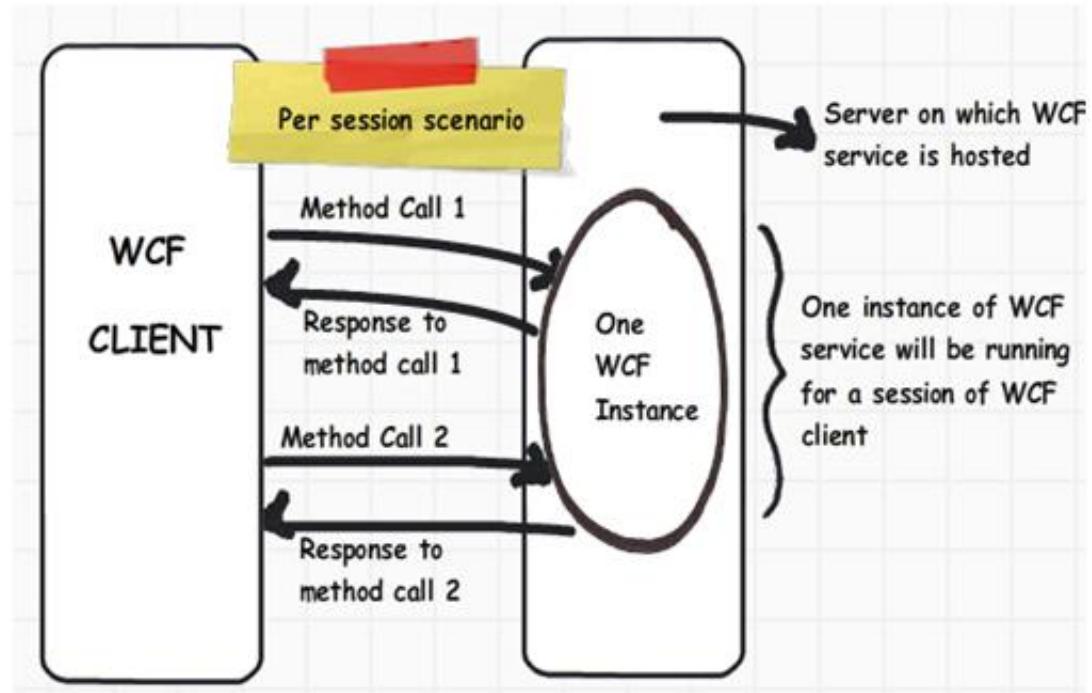


DEMO

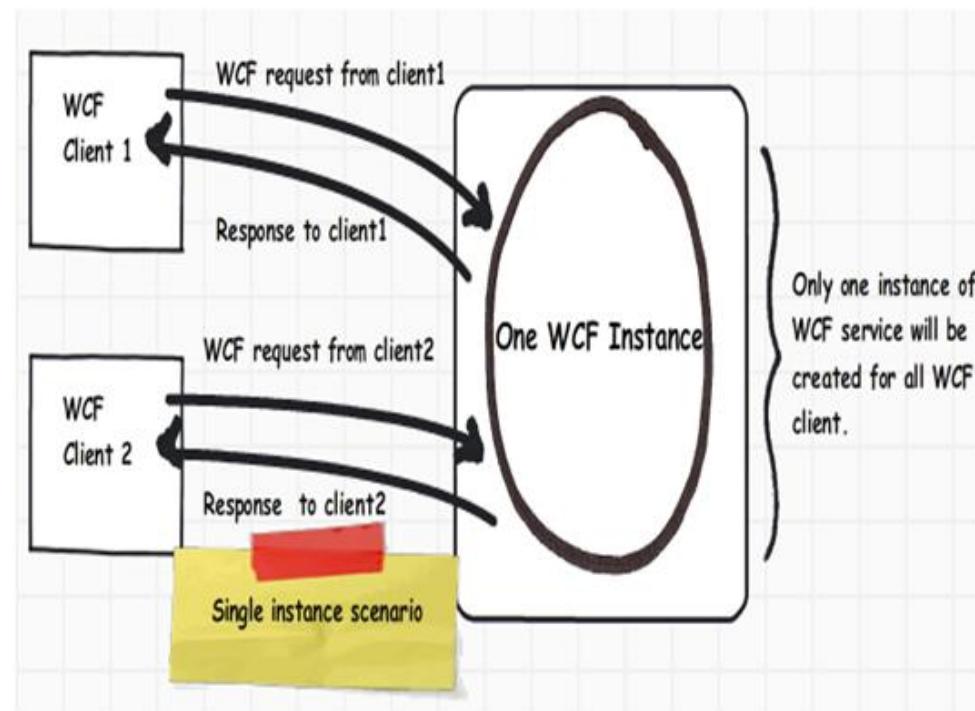
Stateless Service



Stateful Service (i)



Stateful Service (ii)



Outline

- ✓ **Service Oriented Computing**

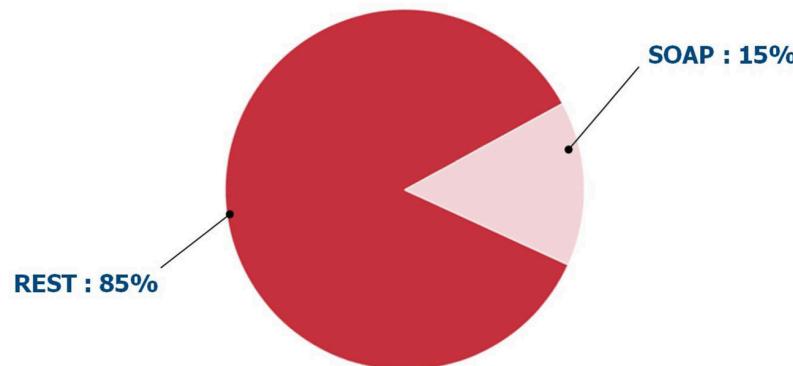
- ✓ Service
- ✓ Services Coordination
- ✓ Service Taxonomy

- **Service Implementation**

- ✓ Web Service
- RESTful Service
- Case study: CouchDB + Deezer

RESTful Service

- Service that conform to the **REST** architecture style
- Offers a simple, interoperable, and flexible way for developing Web applications



► www.oreillynet.com/pub/wlg/3005

REST (i)

- Stands for **Representational State Transfer**
- **Architecture style** defined by Roy T. Fielding in his PhD thesis
 - Describes the architecture of the Web
 - Specifies how Web standards are supposed to be used (**HTTP**, **URIs** and **data formats**)
- **Principles**
 - Every resource has an ID
 - Use standard HTTP operations for interacting with resources
 - Resources have multiple representations
 - Communications is stateless

REST (ii)

- REST is not:
 - A protocol
 - A standard
 - A replacement for Web services (i.e. SOAP)
- **Note:**
 - Using HTTP without following the REST principle is equal of abusing HTTP

Resource

- Key **abstraction** of information, data and operations
 - Everything object (or "thing") in a system can be a resource
- Each resource is **addressable via a URI** (Uniform Resource Identifier)
 - Extensible **naming schema**
 - Works pretty well on **global scale** and is **understood by** practically **everybody**
 - Can be human-readable

`http://example.com/orders/2007/11`

`http://example.com/products?color=green`

URI Schema

- Defines a **set of rules** for identifying a resource
- **Examples**

HTTP

`http://espinosa-oviedo.com`

MAIL

`mailto : javier.espinosa@imag.fr`

GEO

`geo : 48.890172, 2.249922`

Spotify

`spotify : user:javieraespinosa`

Skype

`skype : javiera.espinosa`

LastFM

`lastfm : //user:javieraespinosa`

URI Types

- **Uniform Resource Locator (URL)**
 - Identifies and provides means for locating a resource
- **Uniform Resource Name (URN)**
 - Persistent even if the resource ceases to exist or is unavailable

URL Schema and Syntax (i)

■ URL Schema

- Identifies and provides means for **locating a resource**

scheme :// host [: port] path [? query] [# fragment]

e.g. http :// www.facebook.com : 80 /javiera.espinosa ? sk=info

Allowed characters

0 ... 9 Digit

A ... Z Alphabet

- ... ~ ASCII symbols

Reserved characters

/ ? # [] @ :

! \$ & ‘ () * + , ; =

URL Schema and Syntax (ii)

- Principle

- Encodes **non ASCII-symbols** and **reserved-characters** using the triple %
HEXDIG HEXDIG HEXDIG

- Example

- Hi Zoé ! → Hi %20 Zo %C3 %A9

space é !

Interacting with Resources (i)

- All resources **supports the same API** (i.e. HTTP operations)
 - Each operation has a specific purpose and meaning

Method	Description	Safe	Idempotent
GET	Requests a specific representation of a resource	Yes	Yes
PUT	Create or update a resource with the supplied representation	No	Yes
DELETE	Deletes the specified resource	No	Yes
POST	Submits data to be processed by the identified resource	No	No

- **Note:** The actual semantics of **POST** are defined by the server

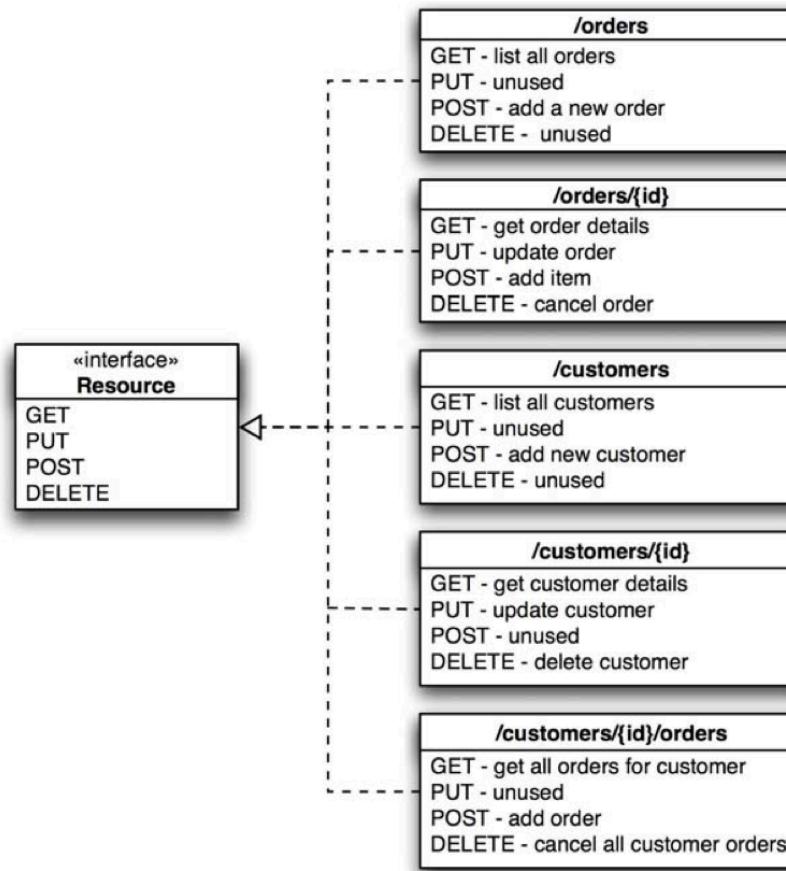
Interacting with Resources (i)

■ HTTP Codes

Status Range	Description	Examples
100	Informational	100 Continue
200	Successful	200 OK
201	Created	
202	Accepted	
300	Redirection	301 Moved Permanently
304	Not Modified	
400	Client error	401 Unauthorized
402	Payment Required	
404	Not Found	
405	Method Not Allowed	
500	Server error	500 Internal Server Error
501	Not Implemented	

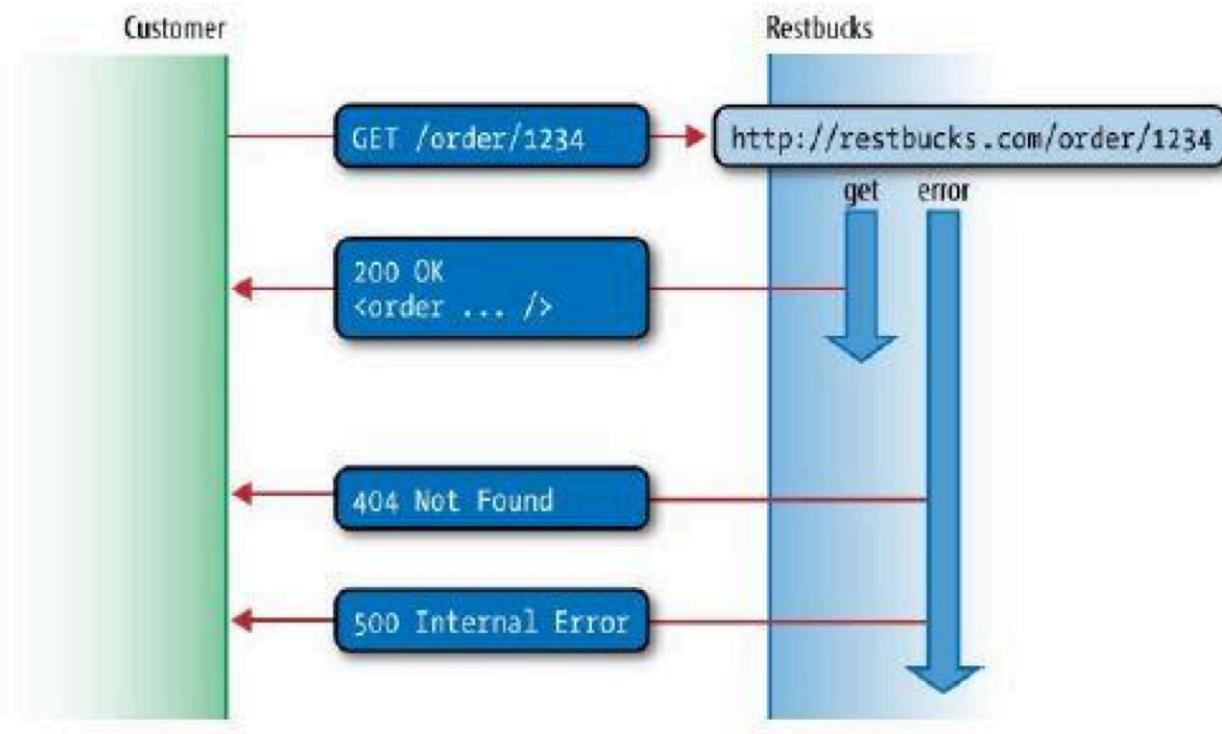
Interacting with Resources (ii)

■ Example



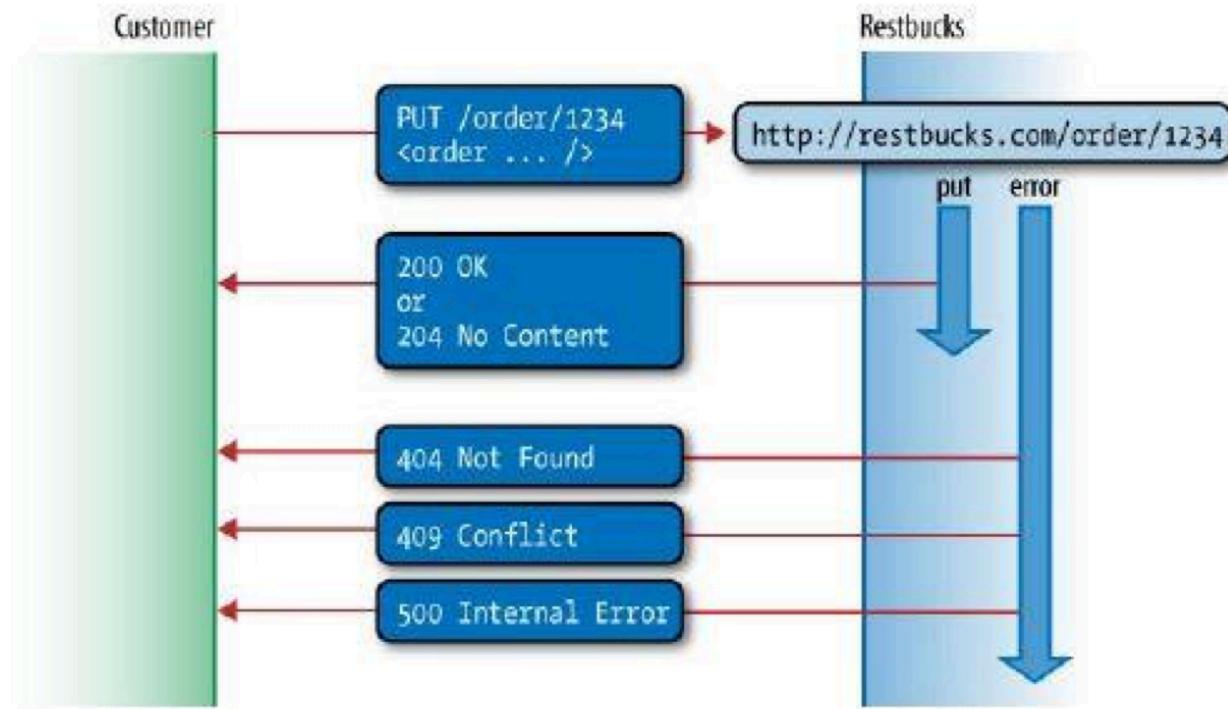
Interacting with Resources (iii)

■ Get a resource



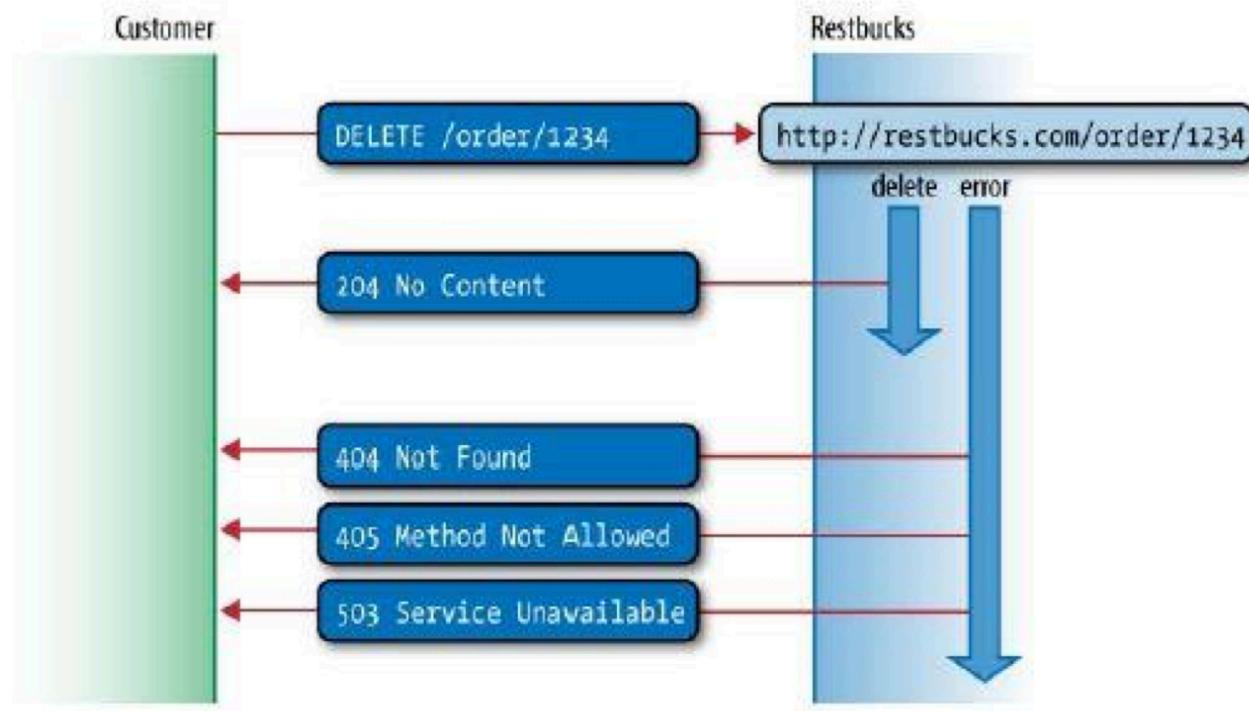
Interacting with Resources (iv)

- Create/Update a resource



Interacting with Resources (v)

- Delete a resource



Representation of Resources

- A resource referenced by one URI can have **different representations**
 - **HTML** (for browsers), **XML** (for application), **JSON** (for JavaScript)

<http://localhost:9999/restapi/books/{id}.xml>

<http://localhost:9999/restapi/books/{id}.json>

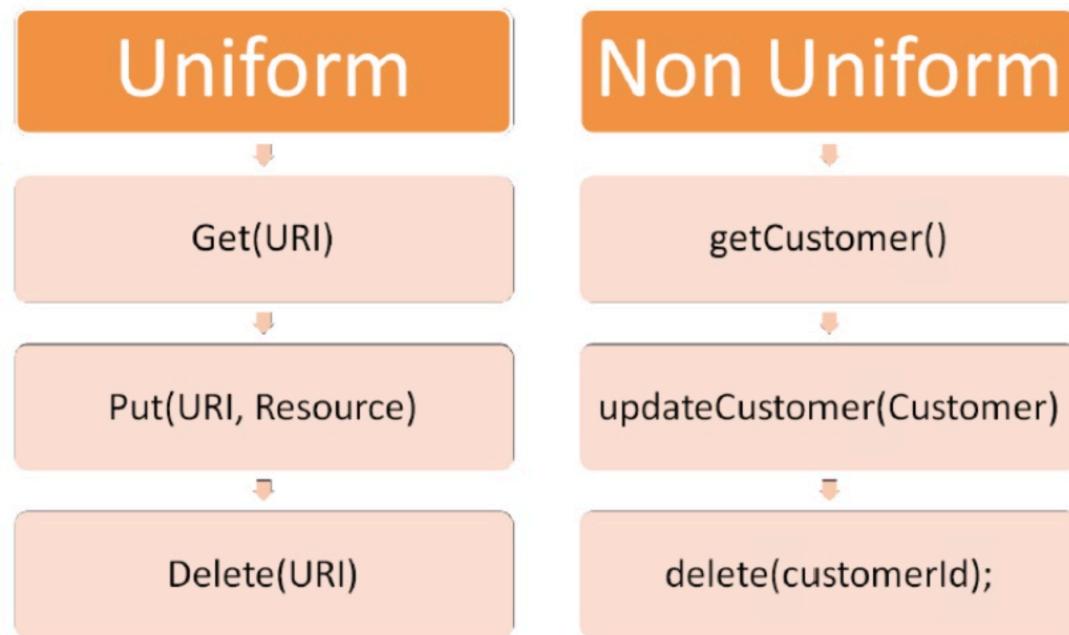
<http://localhost:9999/restapi/books/{id}.pdf>

- If the client "knows" both the HTTP application protocol and a set of data formats then it can interact with any RESTful service in the world

Stateless Communication

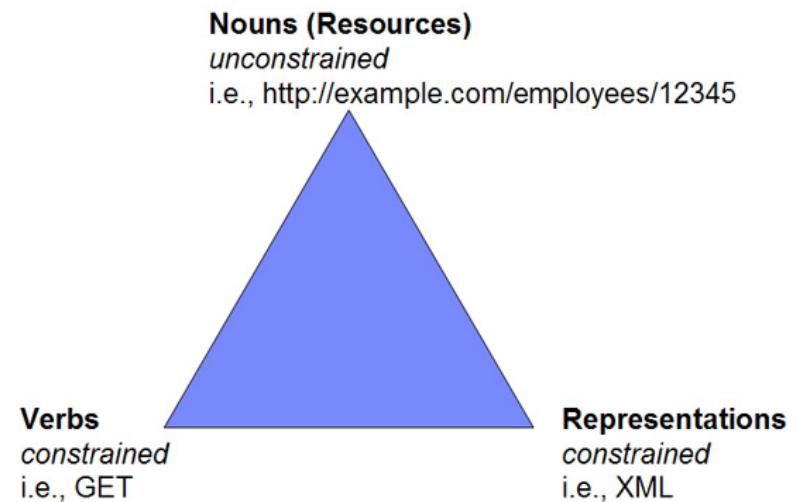
- No client session data stored on the server
- If there are needs for session-specific data, it should be held and maintained by the client and transferred to the server with each request as needed
- A service layer that does not have to maintain client sessions is much easier to scale
 - It isolates the client against changes on the server

Resource Oriented VS Operation Oriented



REST in a Nutshell

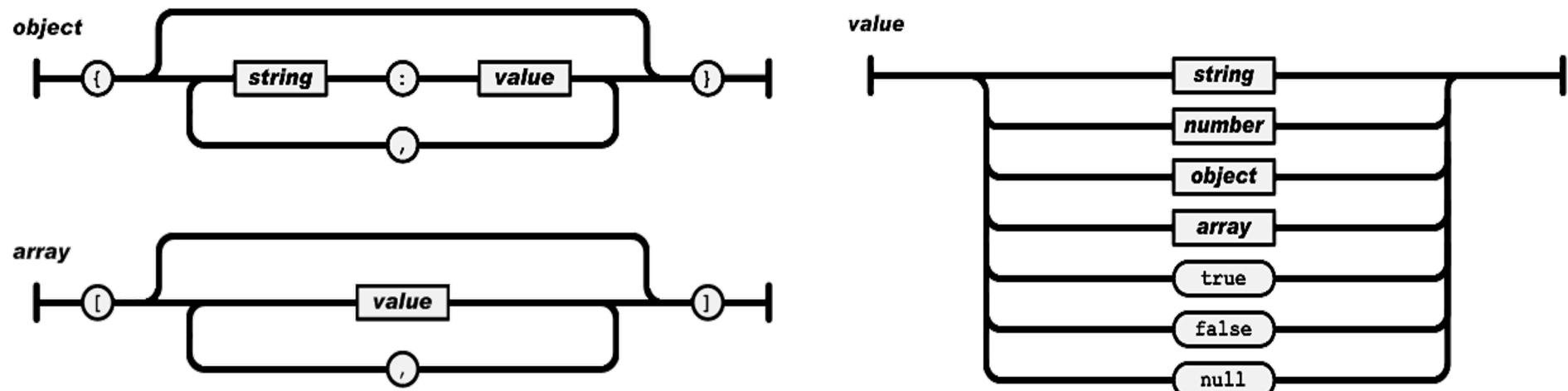
- REST is all about
 - Resources
 - How to manipulate the resource
 - How to represent the resource in different ways



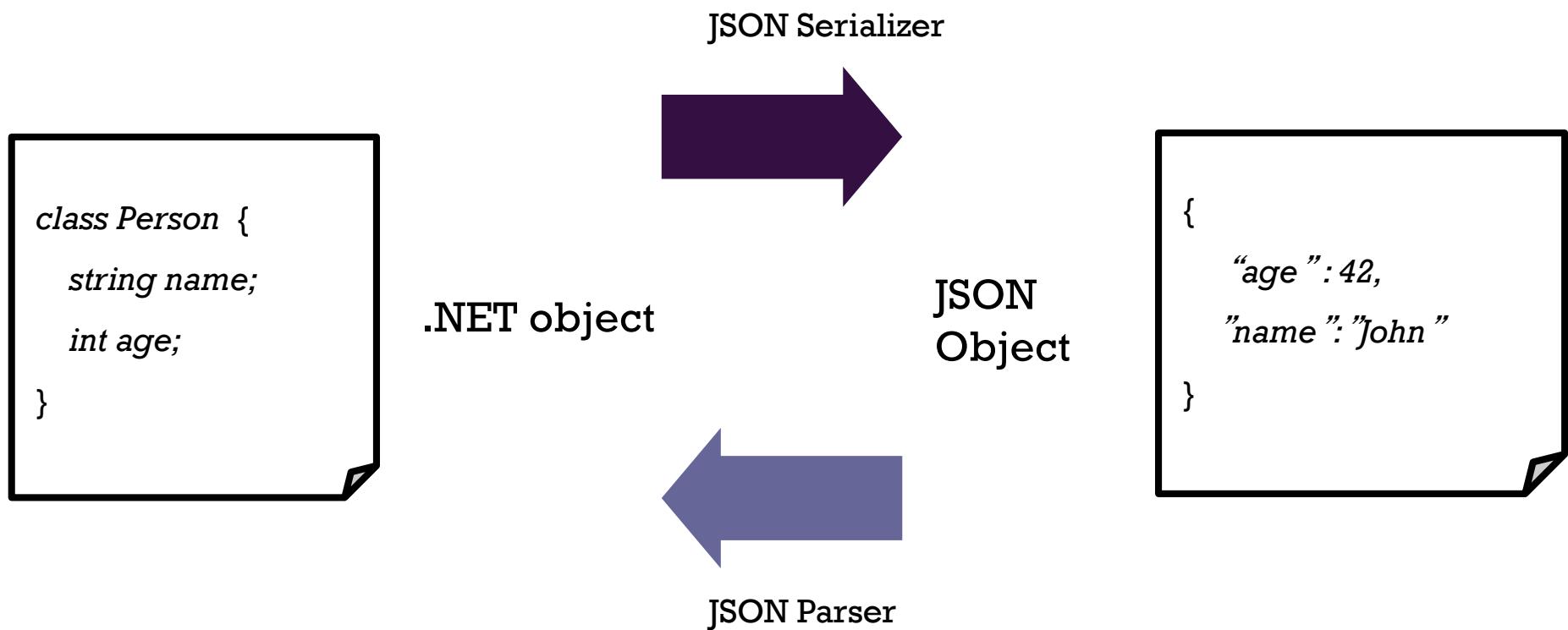
JSON Data Representation

- Lightweight **text format** for interchanging structured data
 - Based on a complex data model
 - Similar to XML
- **Fundamental Concepts**
 - **Object** : unordered set of *name : value* pairs
 - **Array** : ordered set of values
 - **Value** : member of the String, Boolean, Object or Array set

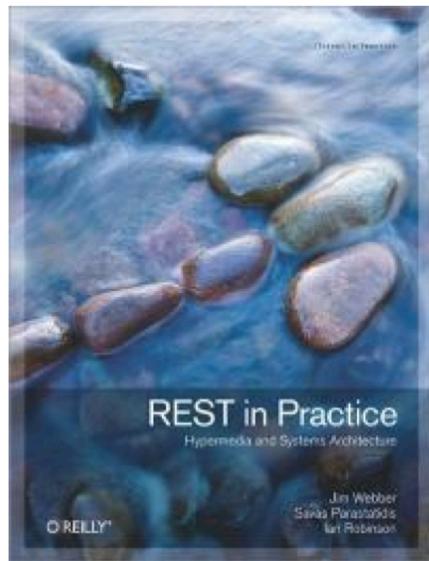
JSON Syntax



JSON Serialization



Books



DEMO

Outline

- ✓ **Service Oriented Computing**

- ✓ Service
- ✓ Services Coordination
- ✓ Service Taxonomy

- **Service Implementation**

- ✓ Web Service
- ✓ RESTful Service
- Case study: CouchDB + Deezer

?

