
RESTful
Seminar on Restful Web Services
Adhin Prakash , 19132406 , 08

Seminar Contents

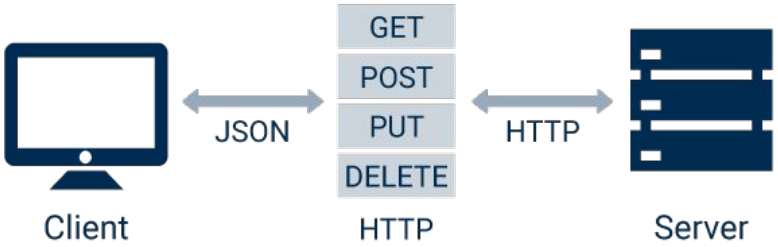
- Introduction.
- What is REST?
- Characteristics of REST.
- Principles of REST.
- REST Architecture.
- SOAP v/s RESTful.
- RESTful Advantages.
- Disadvantages of RESTful.
- Conclus’i’on.

Abstract

- By its nature, user actions within a distributed hypermedia system require the transfer of large amounts of data from where the data is stored to where it is used.
- Thus, the Web architecture must be designed for large-grain data transfer.

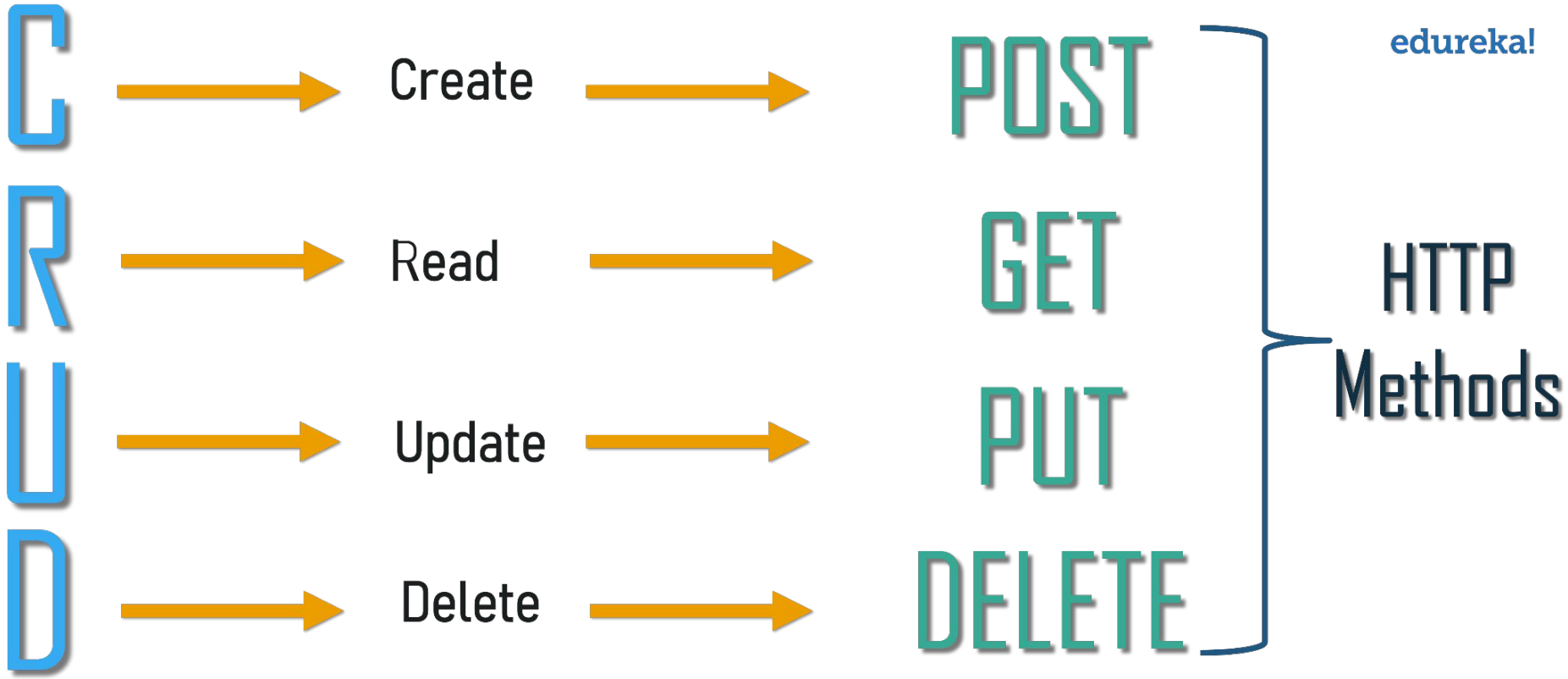
Introduction

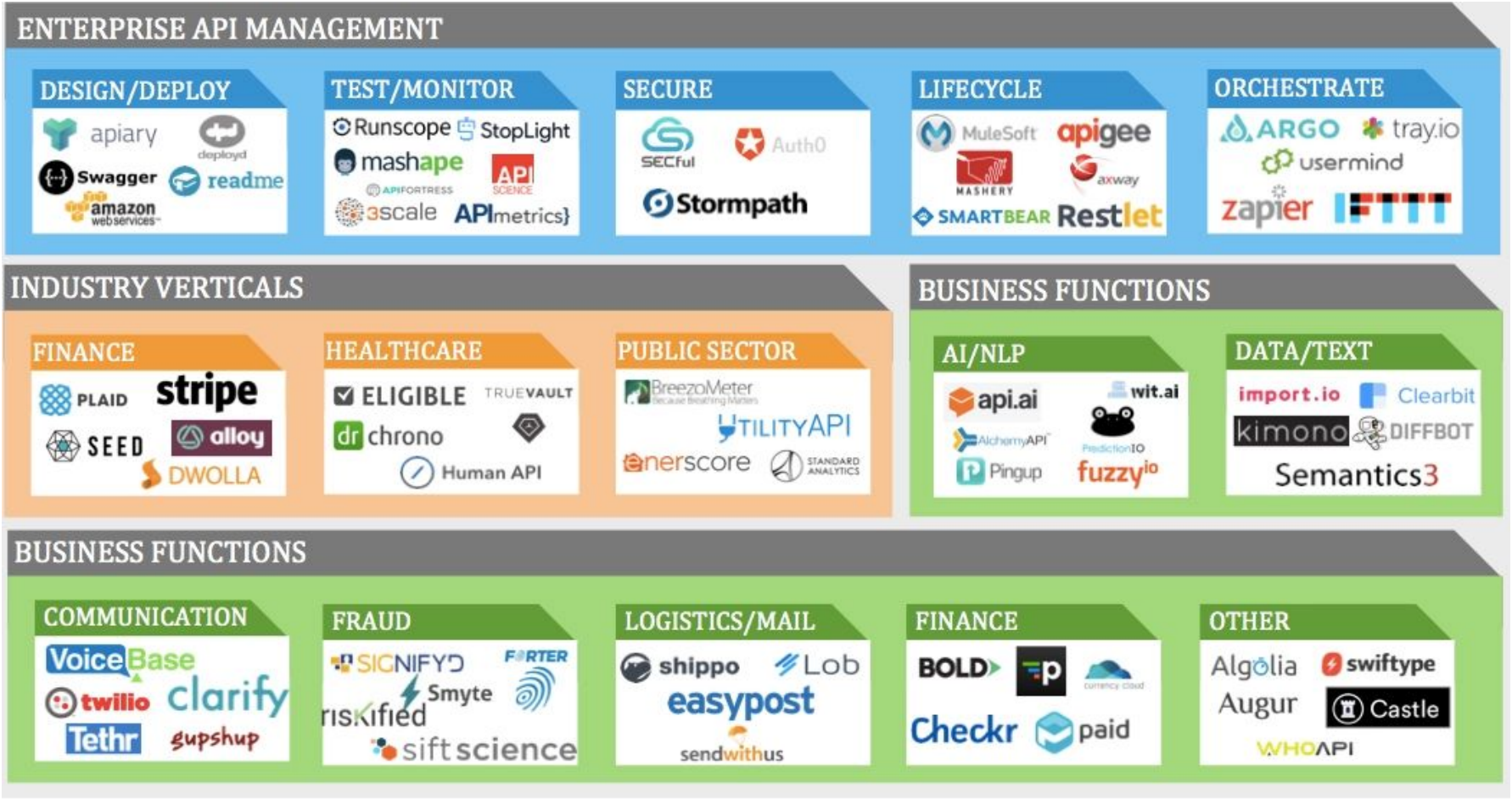
- A RESTFul web services are **based on the HTTP methods** and the **concept of REST**.
- A RESTFul web service typically defines the base URI for the services, the supported MIME-types (**XML, Text, JSON, user-defined,..**) and
- The set of operations (**POST, GET, PUT, DELETE**) which are supported.



What is REST?

- REST defines a set of architectural principles by which you can design Web services that
- focus on a system's resources, including how resource states are addressed and
- transferred over HTTP by a wide range of clients written in different languages.


C
R
U
D
Create
Read
Update
Delete
POST
GET
PUT
DELETE
HTTP
Methods
edureka!


ENTERPRISE API MANAGEMENT
DESIGN/DEPLOY
apiary
deployd
Swagger
readme
amazon web services
TEST/MONITOR
Runscope
StopLight
mashape
APIFORTRESS
API SCIENCE
3scale
APImetrics}
SECURE
SECful
Auth0
Stormpath
LIFECYCLE
MuleSoft
apigee
MASHERY
axway
SMARTBEAR
Restlet
ORCHESTRATE
ARGO
tray.io
usermind
zapier
IFTTT
INDUSTRY VERTICALS
FINANCE
PLAID
stripe
SEED
alloy
DWOLLA
HEALTHCARE
ELIGIBLE
TRUEVAULT
dr chrono
Human API
PUBLIC SECTOR
BreezoMeter
UTILITYAPI
enerscore
STANDARD ANALYTICS
BUSINESS FUNCTIONS
AI/NLP
api.ai
wit.ai
AlchemyAPI
PredictionIO
Pingup
fuzzyio
DATA/TEXT
import.io
Clearbit
kimono
DIFFBOT
Semantics3
BUSINESS FUNCTIONS
COMMUNICATION
VoiceBase
twilio
clarify
Tethr
gupshup
FRAUD
SIGNIFYD
FORTER
Smyte
riskified
sift science
LOGISTICS/MAIL
shippo
Lob
easypost
sendwithus
FINANCE
BOLD
currency cloud
Checkr
paid
OTHER
Algolia
swiftype
Augur
Castle
WHOAPI

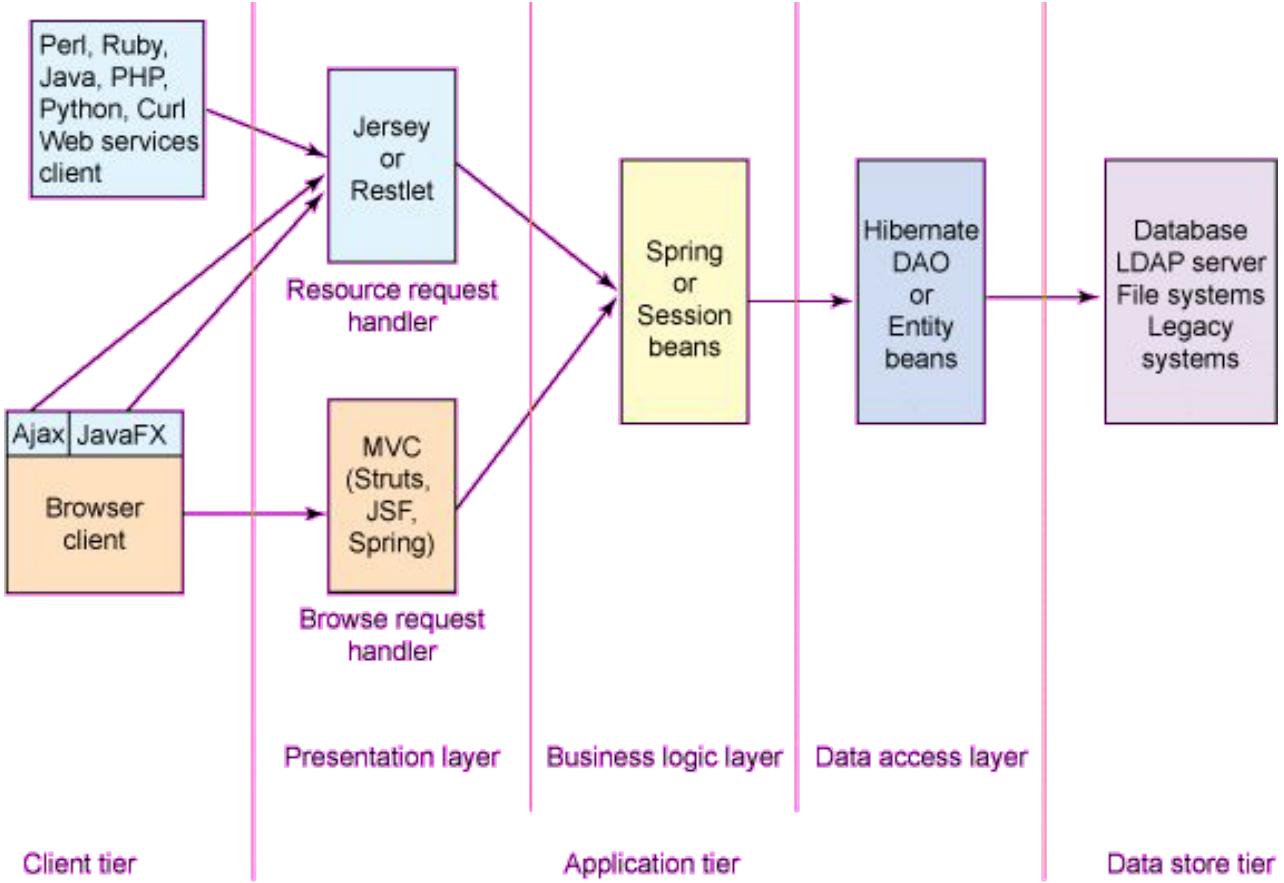
REST Web Services Characteristics

- **Client-Server**: a pull-based interaction style: consuming components pull representations.
- **Named resources** - the system is comprised of resources which are named using a URL.
- **Uniform interface**: all resources are accessed with a generic interface (e.g., **HTTP GET, POST, PUT, DELETE**).

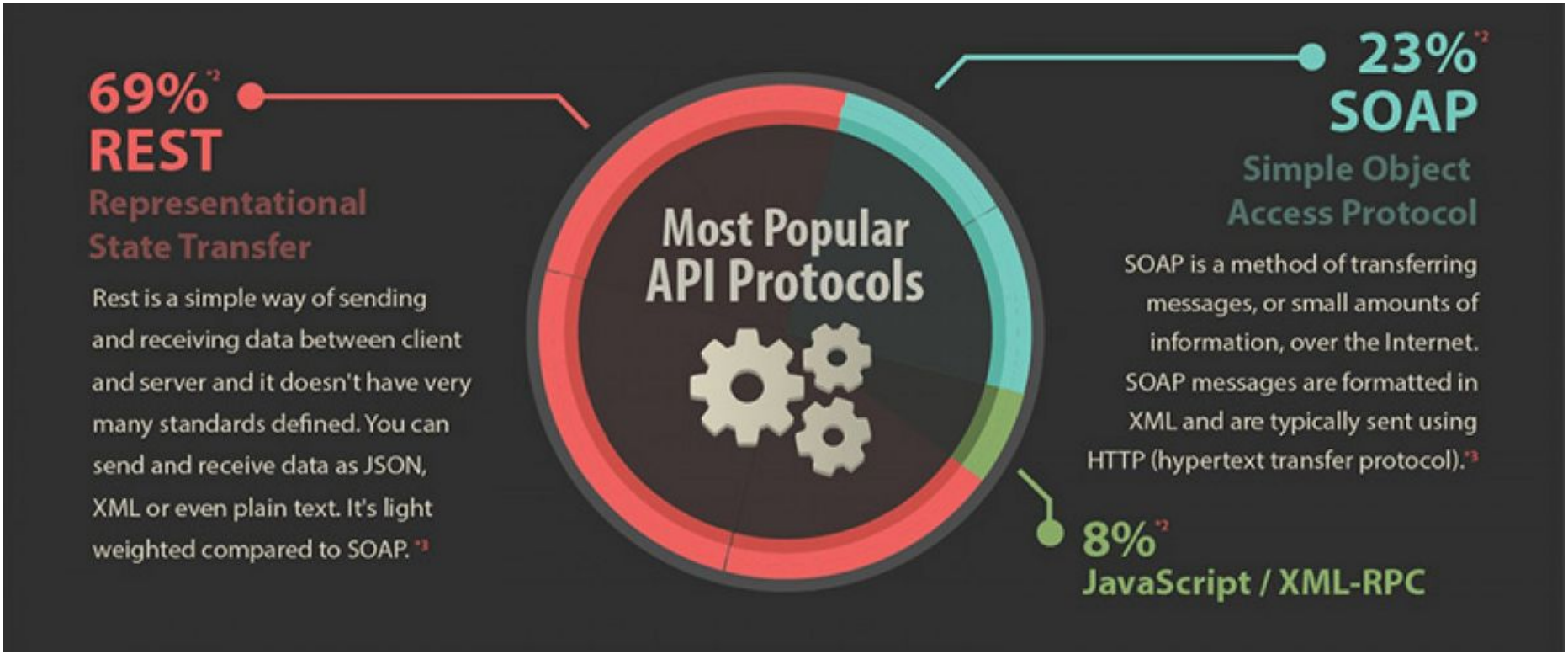
Principles of REST Web Service

- Describe **how your services are to be invoked** using either a **WSDL document**, or simply an **HTML document**.
- **Categorize** your resources according to whether clients can just receive a representation of the resource,
- or whether clients can **modify (add to)** the resource.

RESTful Web Services Architecture



SOAP v/s RESTful


69%[2]
REST
Representational State Transfer
Rest is a simple way of sending and receiving data between client and server and it doesn't have very many standards defined. You can send and receive data as JSON, XML or even plain text. It's light weighted compared to SOAP.[3]
Most Popular API Protocols
23%[2]
SOAP
Simple Object Access Protocol
SOAP is a method of transferring messages, or small amounts of information, over the Internet. SOAP messages are formatted in XML and are typically sent using HTTP (hypertext transfer protocol).[3]
8%[2]
JavaScript / XML-RPC

REST Design Guidelines

- Queries should not return an overload of data. If needed, provide a paging mechanism. For example, a "product list".
- GET request should return the first n products (e.g., the first 10), with next/prev links.

A few Tests of RESTfulness

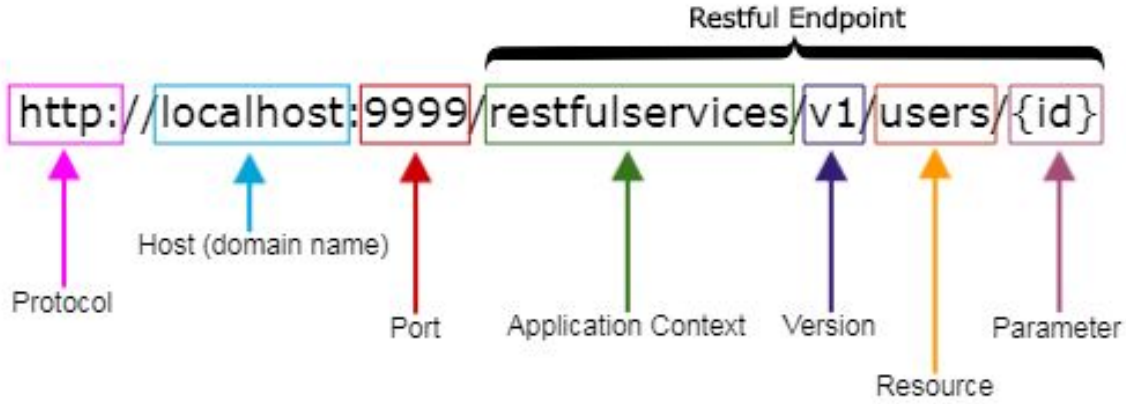
- Can I do a GET on the URLs that I POST to?
- If so, do I get something that in some way represents the state of what I've been building up with the POSTs?
- HTML forms almost always fail miserably.

Advantages

- **Scalable** component interactions
- **General** interfaces
- **Independently** deployed connectors.
- Reduced interaction **Latency**.
- Strengthened **Security**.
- **Safe Encapsulation** of legacy systems.

..Advantages

- Separates server implementation from the client's perception of resources ("**Cool URIs Don't Change**").
- Scales well to **large numbers of clients**.
- Enables **transfer of data in streams** of **unlimited size** and **type**.



Disadvantages

- It sacrifices some of the advantages of other architectures.
- Stateful interaction with an FTP site.
- It retains a **single interface for everything**.

Conclus'i'on

- **Service-Oriented Architecture** can be implemented in different ways.
- General focus is on whatever architecture gets the job done.
- The decision of which to use depends entirely on the circumstances of the application.

Thanksss