

INTRODUCTION TO WEB SERVICES

In28Minutes

Spring Web Services

- 1** **Introduction To Web Services**
- 2** **Spring Framework in 10 Steps**
- 3** **Spring Boot in 10 Steps**
- 4** **SOAP Web Services - 18 Steps**
- 5** **RESTful Web Services - 35 Steps**
- 6** **JPA in 10 Steps**

In28Minutes

Spring Web Services

Web Services

What?

Why?

How?

When?

Types

REST

SOAP

SOAP vs REST

REST Web Services

Validation

Exceptions

HATEOAS

Versioning

Swagger

Filtering

SOAP

WSDL

XSD

JAXB

EndPoint

WS Security

Spring

Spring Boot

JPA

Maven

Postman

Wizdler

Tools/Frameworks

Dynamic
Autowiring

6

Types of
Autowiring

7

Spring
Modules

8

Spring
Projects

9

Why is Spring
Popular?

10

1

Setting up a
Spring Project

2

Understand
Tight coupling

3

Introduce
loose coupling

4

Using Spring
to manage
beans

5

What's
happening in
Background?

In28Minutes

Spring

Framework

In28Minutes

Spring Boot

Introduction

Goals

Features

Spring Initializr

Spring vs Spring MVC vs Spring Boot

Starters

Starter Web

Starter JPA

Features

Auto Configuration

Developer Tools

Actuator



Course Management

Retrieve

RetrieveAll

Delete

XSD

JAXB

WSDL

EndPoint

Concepts

SOAP

Headers

Body

Fault

Security

WS Security

Tools/Frameworks

Spring

Spring Boot

Wizdler

In28Minutes

RESTful Web Services

Basics

HTTP Request Methods

GET

POST

DELETE

Exception Handling

Validation

HATEOAS

HTTP Response Status

200

400

404

Versioning

Swagger

Filtering

Monitoring

Content Negotiation

Internationalization

Spring

Spring Boot

JPA

Maven

Postman



Introduction

History

JDBC

Spring JDBC

myBatis

JPA Concepts

Mappings

Entity

Entity Manager

Frameworks

Hibernate

Spring Data

Spring Data JPA

Spring Boot

In Memory Database H2

WEB SERVICE

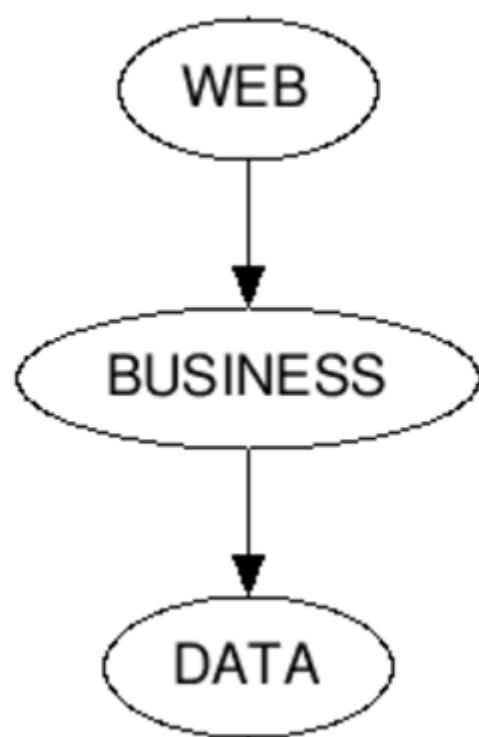
Service delivered over the web?

Your todos are

Description	Target Date	Is it Done?
Learn Spring MVC	28/01/2017	false
Default Desc fskdjfkklasdjf	26/01/2017	false
Learn Struts	24/01/2017	false
Default Desc	24/01/2017	false
Learn Hibernate 2	24/01/2017	false

Is the Todo Management Application a Web Service?

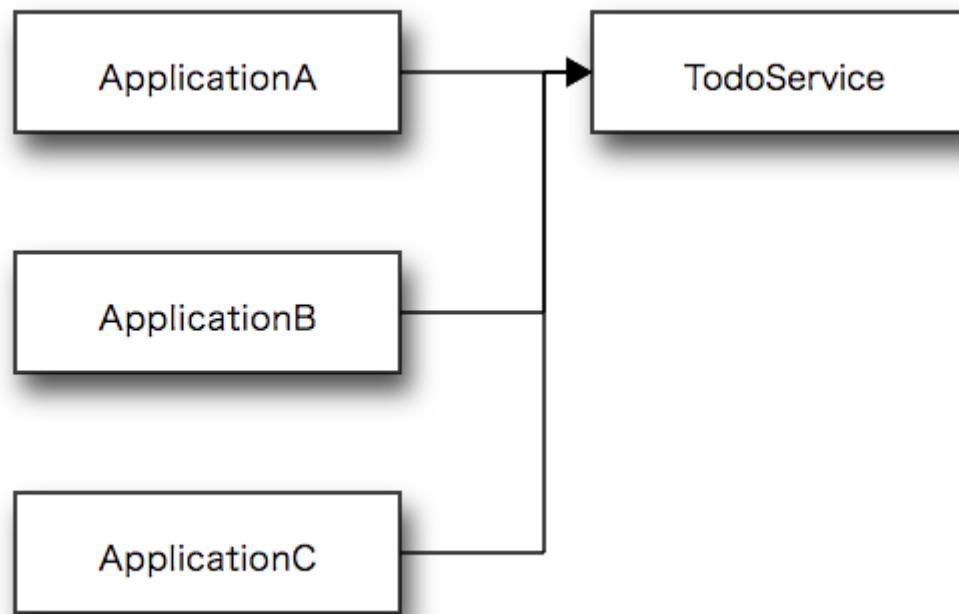
- It delivers HTML output - Not consumable by other applications.



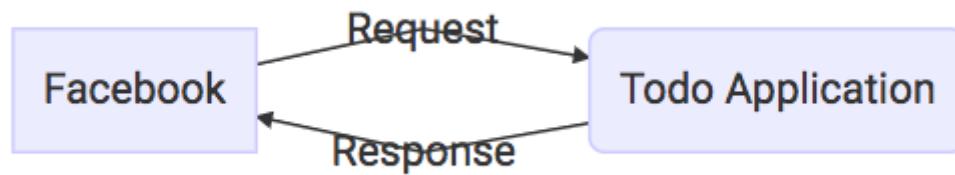
- Can I reuse the Business Layer by creating a JAR?
 - Not Platform independent
 - Communication of Changes
 - Managing Dependencies - like Database

*How can I make my Todo application
consumable by other applications?*

*That where we get into the concept of a
web service!*







WEB SERVICE - W3C DEFINITION

*Software system designed to support
interoperable machine-to-machine
interaction over a network.*

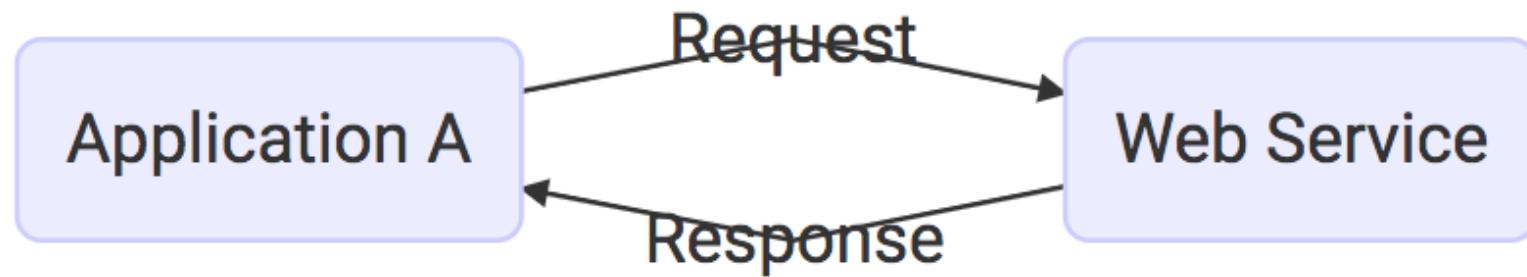
3 KEYS

- Designed for machine-to-machine (or application-to-application) interaction
- Should be interoperable - Not platform dependent
- Should allow communication over a network

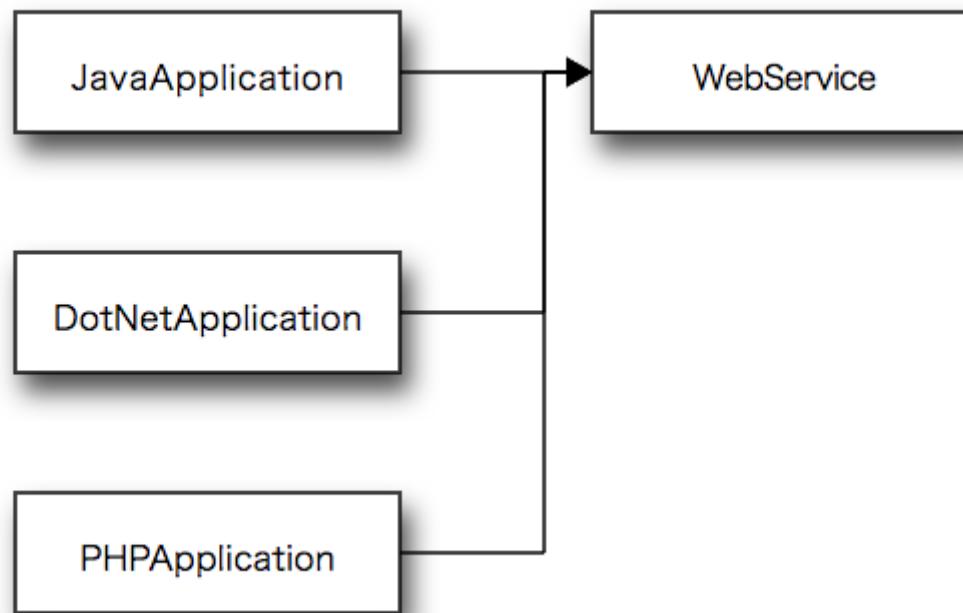
HOW?

*How does data exchange between
applications take place?*





*How can we make web services
platform independent?*



extensible mark up language



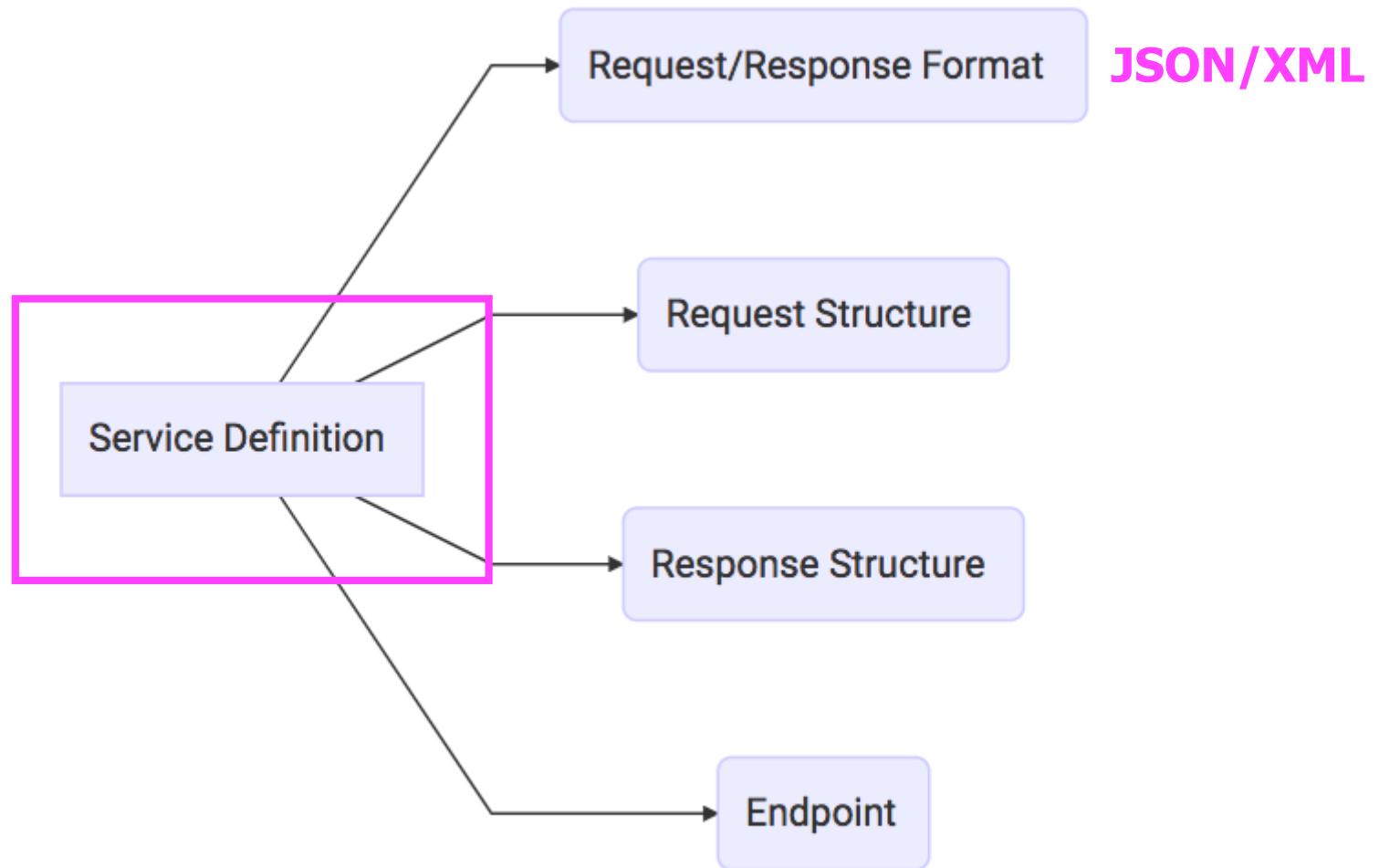
```
<getCourseDetailsRequest>
    <id>Course1</id>
</getCourseDetailsRequest>
```

JSON

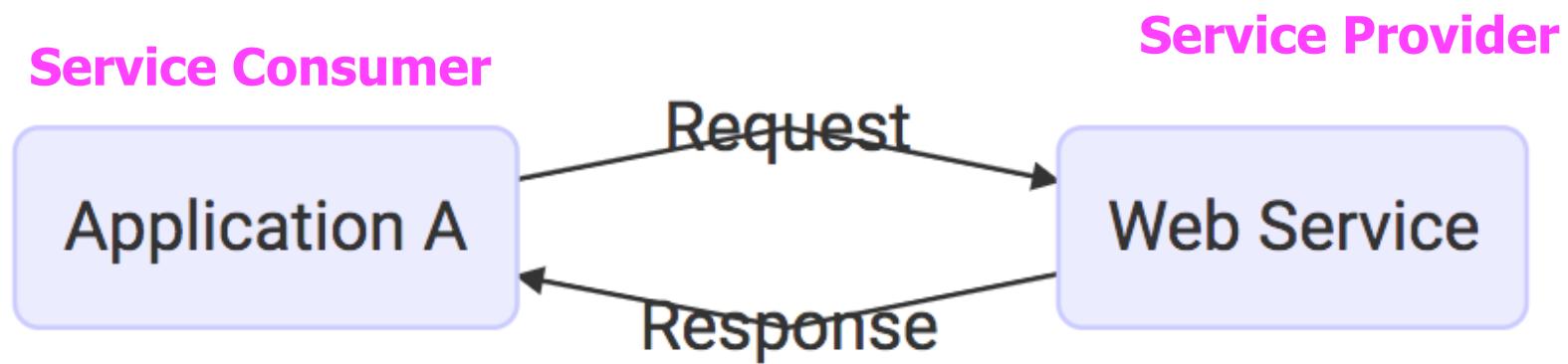
Java script Object Notation

```
[  
  {  
    "id": 1,  
    "name": "Even",  
    "birthDate": "2017-07-10T07:52:48.270+0000"  
  },  
  {  
    "id": 2,  
    "name": "Abe",  
    "birthDate": "2017-07-10T07:52:48.270+0000"  
  }  
]
```

*How does the Application A know the
format of Request and Response?*



*How does Application A and Web
Service convert its internal data to (XML
or JSON)?*



KEY TERMINOLOGY

- Request and Response
- Message Exchange Format
 - XML and JSON

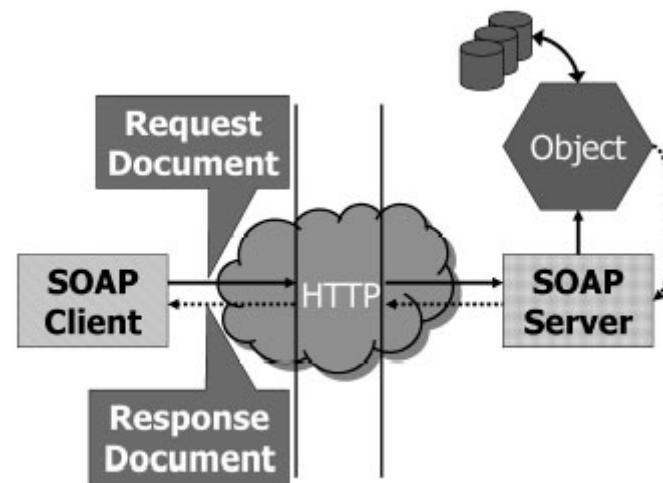
KEY TERMINOLOGY

- Service Provider or Server
- Service Consumer or Client
- Service Definition

KEY TERMINOLOGY

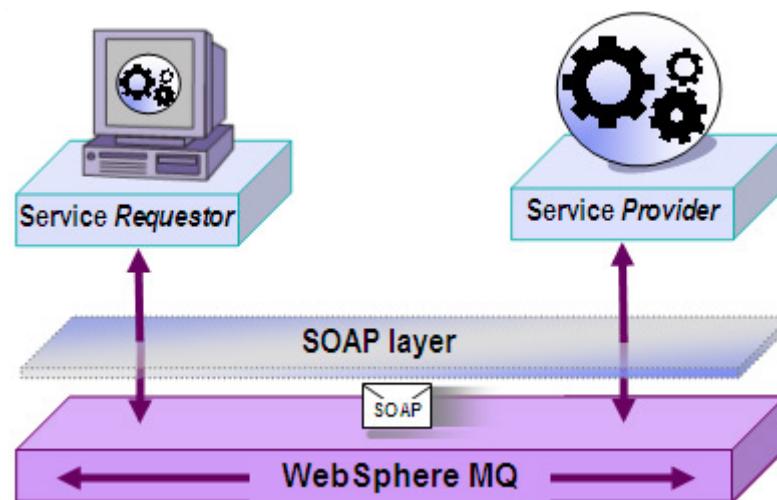
- Transport
 - HTTP and MQ

Message Queue



HTTP

Communication happens through Web



Message Queue

Communication happens through Queue

WEB SERVICE GROUPS

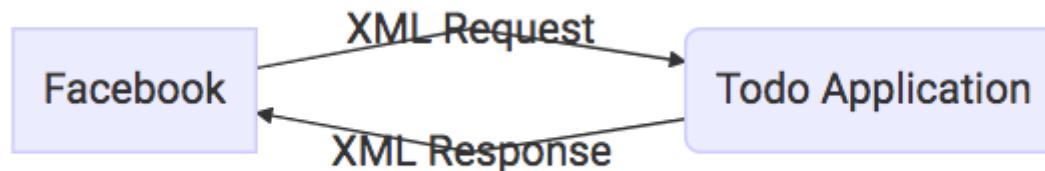
- SOAP-based
- REST-styled

*SOAP and REST are not really
comparable.*

SOAP

SOAP?

Simple Object Access Protocol
只能用xml



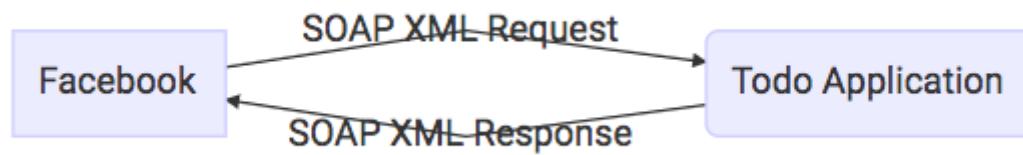
```
<getCourseDetailsRequest>
    <id>Course1</id>
</getCourseDetailsRequest>
```

SOAP-ENV: Envelope

SOAP-ENV: Header

Header is optional

SOAP-ENV: Body



```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
    <ns2:getCourseDetailsResponse xmlns:ns2="http://in28mi
        <ns2:course>
            <ns2:id>Course1</ns2:id>
            <ns2:name>Spring</ns2:name>
            <ns2:description>10 Steps</ns2:description>
        </ns2:course>
    </ns2:getCourseDetailsResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP

- Format 只能用xml
 - SOAP XML Request
 - SOAP XML Response
- Transport
 - SOAP over MQ
 - SOAP over HTTP
- Service Definition
 - WSDL

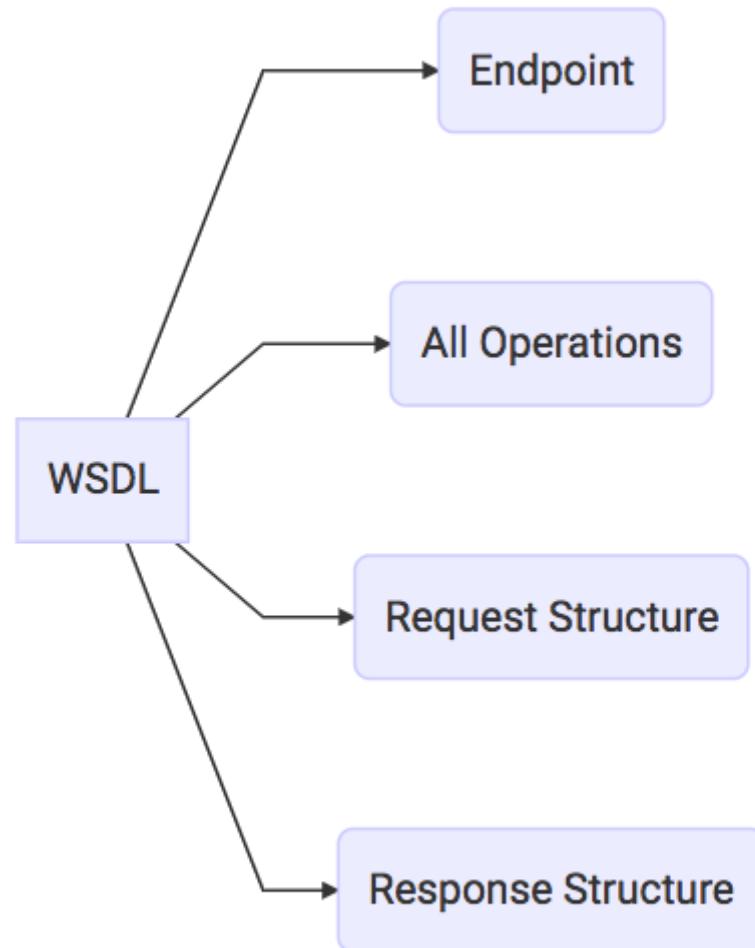
SOAP: Simple Object Access Protocol - xml
WSDL: Web Service Definition Language

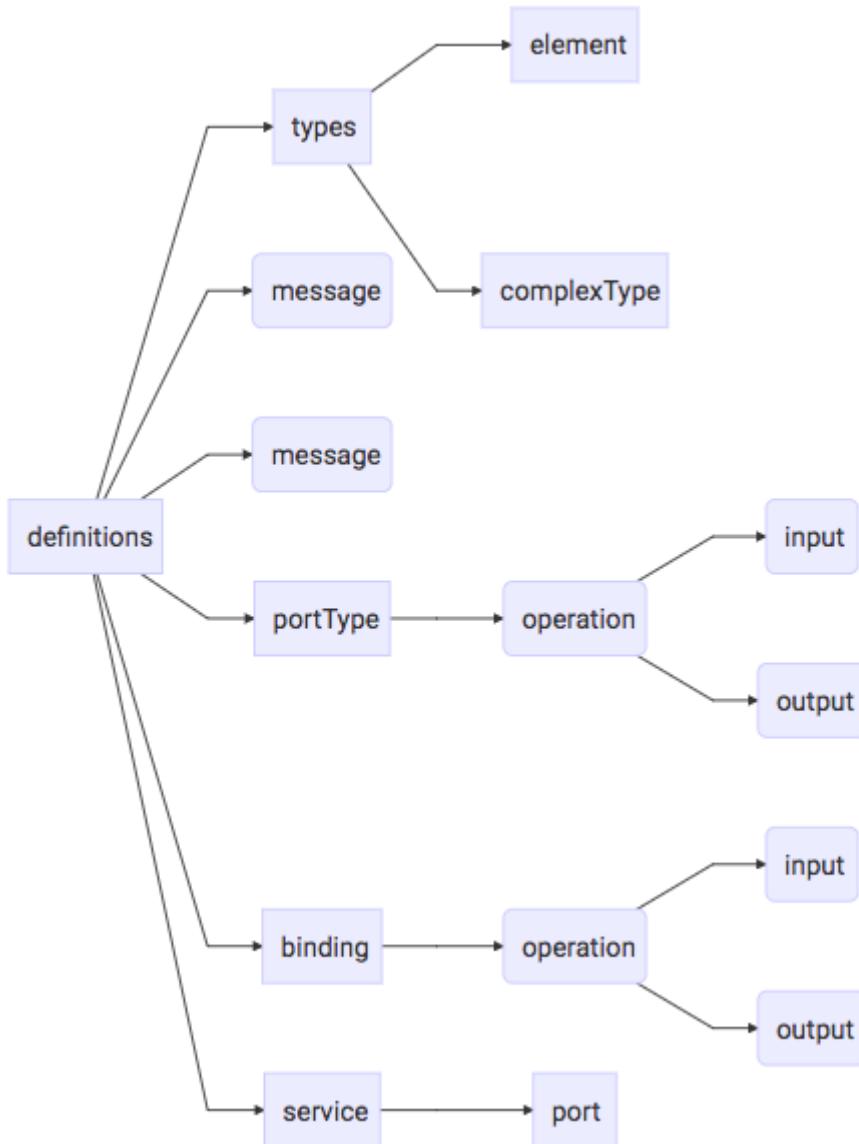
WSDL

SOAP: Simple Object Access Protocol
WSDL: Web Service Definition Language

SOAP: Simple Object Access Protocol

WSDL: Web Service Definition Language





In28Minutes

SOAP Web Service

Service
Definition

XML

XSD

WSDL

XML Java
Binding

JAXB

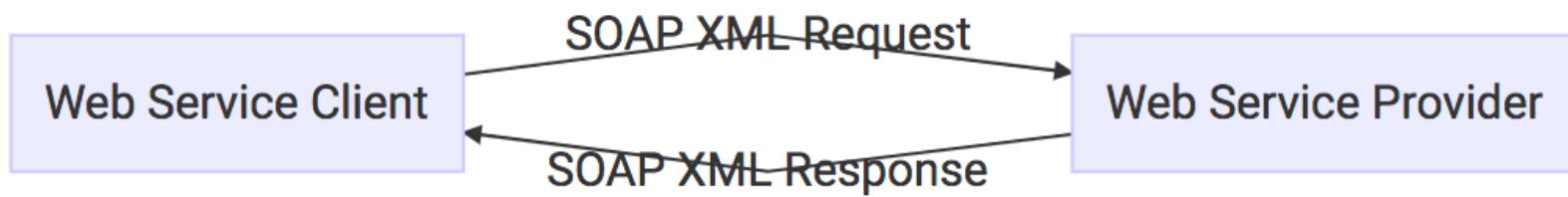
Framework
Configuration

Endpoint

WSDL-Generation

SOAP Web
Service Client

Wizdler Chrome Plugin



REST

REpresentational State Transfer

REpresentational State Transfer

*REST is a style of software architecture
for distributed hypermedia systems*

REST: REpresentational State Transfer

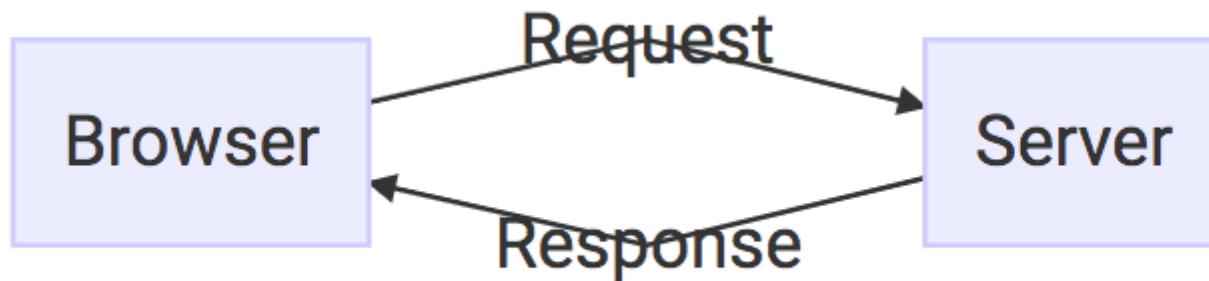
MAKE BEST USE OF HTTP

REST(REpresentational State Transfer)

HTTP **Hyper Text Transfer Protocol**

HTTP Methods (GET, PUT, POST..)

HTTP Status Codes (200, 404..)



URL: Uniform Resource Locator

KEY ABSTRACTION - RESOURCE

- A resource has an URI (Uniform Resource Identifier)
 - /users/Ranga/todos/1
 - /users/Ranga/todos
 - /users/Ranga
- A resource can have different representations
 - XML
 - HTML
 - JSON

Fact of REST: think in terms of Resources
=> **Verbs + Resources**

EXAMPLE

- Create a User - POST /users
- Delete a User - DELETE /users/1
- Get all Users - GET /users
- Get one Users - GET /users/1

REST: REpresentational State Transfer

REST

- Data Exchange Format
 - No Restriction. JSON is popular
- Transport
 - Only HTTP
- Service Definition
 - No Standard. WADL/Swagger/...

SAOP: format of xml

- can only use xml
- MQ or HTTP

REST: software architecture style

- xml or JSON
- HTTP only

REST VS SOAP

- Restrictions vs Architectural Approach
- Data Exchange Format
- Service Definition
- Transport
- Ease of implementation

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/  
    <SOAP-ENV:Header/>  
    <SOAP-ENV:Body>  
        <ns2:GetCourseDetailsRequest xmlns:ns2="http://in28min  
            <ns2:id>Course1</ns2:id>  
        </ns2:GetCourseDetailsRequest>  
    </SOAP-ENV:Body>  
</SOAP-ENV:Envelope>
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
    <ns2:GetCourseDetailsResponse xmlns:ns2="http://in28mi
        <ns2:CourseDetails>
            <ns2:id>Course1</ns2:id>
            <ns2:name>Spring</ns2:name>
            <ns2:description>10 Steps</ns2:description>
        </ns2:CourseDetails>
    </ns2:GetCourseDetailsResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

RICHARDSON MATURITY MODEL

LEVEL 0

EXPOSE SOAP WEB SERVICES IN REST STYLE

- <http://server/getPosts>
- <http://server/deletePosts>
- <http://server/doThis>

LEVEL 1

EXPOSE RESOURCES WITH PROPER URI

- <http://server/accounts>
- <http://server/accounts/10>

**NOTE : IMPROPER USE OF HTTP
METHODS**

LEVEL 2

LEVEL 1 + HTTP METHODS

LEVEL 3

LEVEL 2 + HATEOAS

DATA + NEXT POSSIBLE ACTIONS

BEST PRACTICES IN RESTFUL DESIGN

CONSUMER FIRST

MAKE BEST USE OF HTTP

REQUEST METHODS

- GET
- POST
- PUT
- DELETE

RESPONSE STATUS

- 200 - SUCCESS
- 404 - RESOURCE NOT FOUND
- 400 - BAD REQUEST
- 201 - CREATED
- 401 - UNAUTHORIZED
- 500 - SERVER ERROR

**NO SECURE INFO IN
URI**

USE PLURALS

- Prefer /users to /user
- Prefer /users/1 to /user/1

**USE NOUNS FOR
RESOURCES**

FOR EXCEPTIONS

DEFINE A CONSISTENT
APPROACH

- /search
- PUT /gists/{id}/star
- DELETE /gists/{id}/star

CONSUMER FIRST

ARCHITECTURAL STANDARDS

DEFINE ORGANIZATIONAL STANDARDS

- YARAS - <https://github.com/darrin/yaras>

NAMING RESOURCES

REQUEST RESPONSE STRUCTURES

COMMON FEATURES STANDARDIZATION

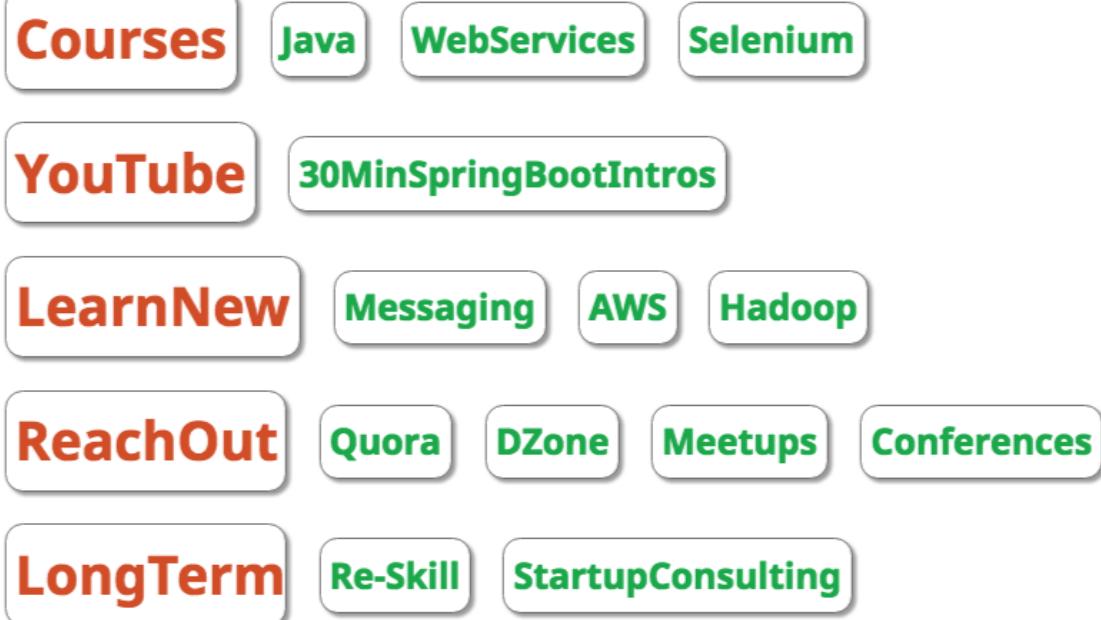
- Error Handling
- Versioning
- Searching
- Filtering
- Support for Mock Responses
- HATEOAS

BUILD A FRAMEWORK

FOCUS ON DECENTRALIZED GOVERNANCE

In28Minutes

My Focus Areas



In28Minutes
Learning
RoadMap

