

INTRODUCTION TO WEB SERVICES

WEB SERVICE

Service delivered over the web?

localhost

in28minutes/SpringBootWebApplicationStepByStep:...SpringBootWebApplicationStepByStep/Step21.md at...SpringMvcStepByStep/Step37.md at master · in28...First Web Application

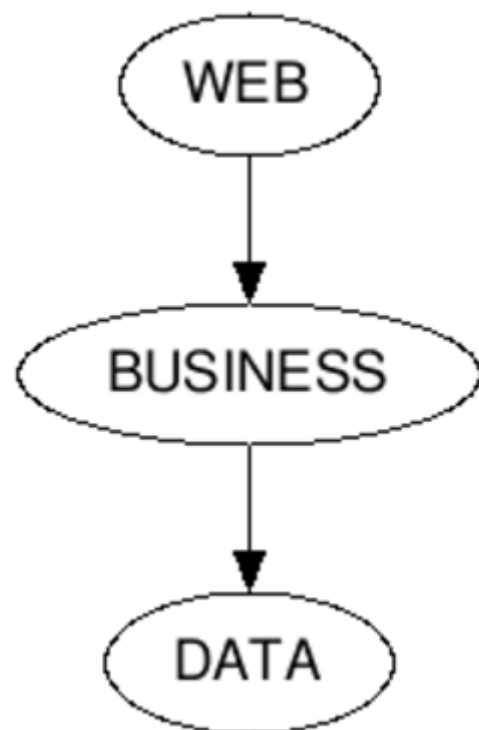
in28MinutesHomeTodos

Your todos are

| Description | Target Date | Is it Done? | | |
|---------------------------|-------------|-------------|--------|--------|
| Learn Spring MVC | 28/01/2017 | false | Update | Delete |
| Default Desc fskdjflksdjf | 26/01/2017 | false | Update | Delete |
| Learn Struts | 24/01/2017 | false | Update | Delete |
| Default Desc | 24/01/2017 | false | Update | Delete |
| Learn Hibernate 2 | 24/01/2017 | false | Update | Delete |

*Is the Todo Management Application a
Web Service?*

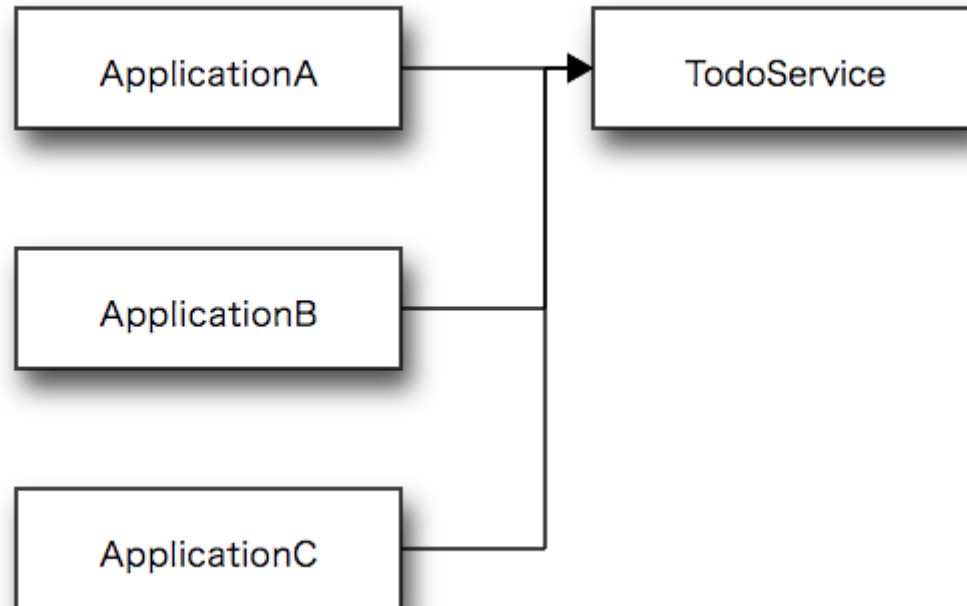
- It delivers HTML output - Not consumable by other applications.



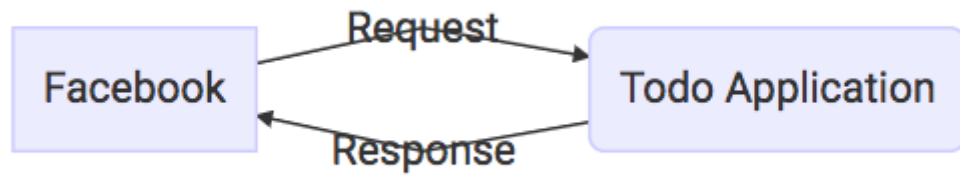
- Can I reuse the Business Layer by creating a JAR?
 - Not Platform independent
 - Communication of Changes
 - Managing Dependencies - like Database

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

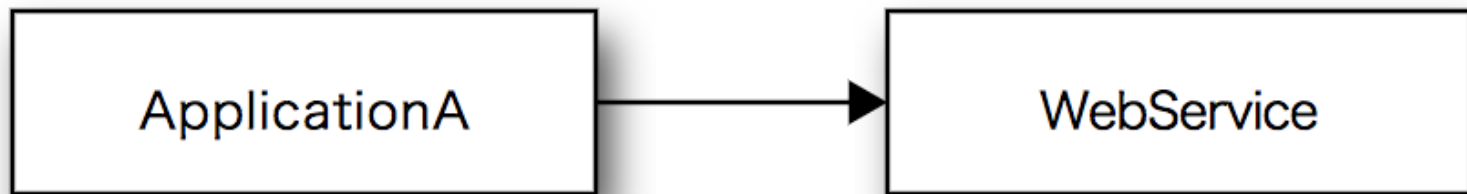
*Software system designed to support
interoperable machine-to-machine
interaction over a network.*

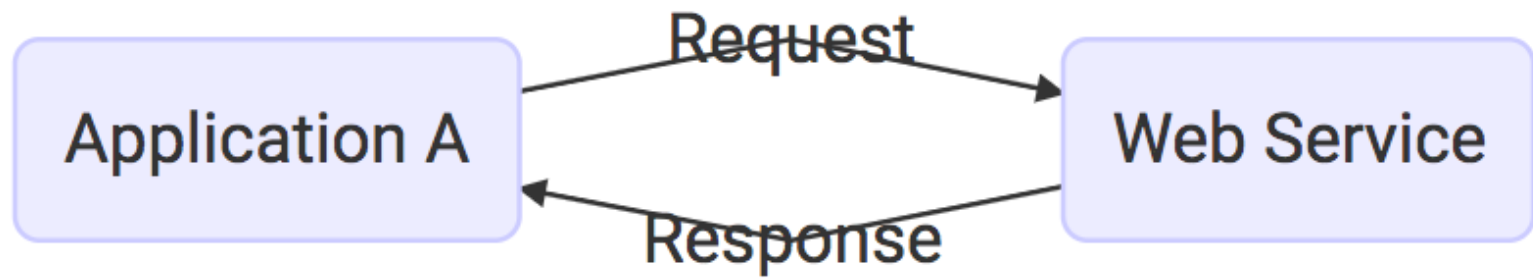
3 KEYS

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

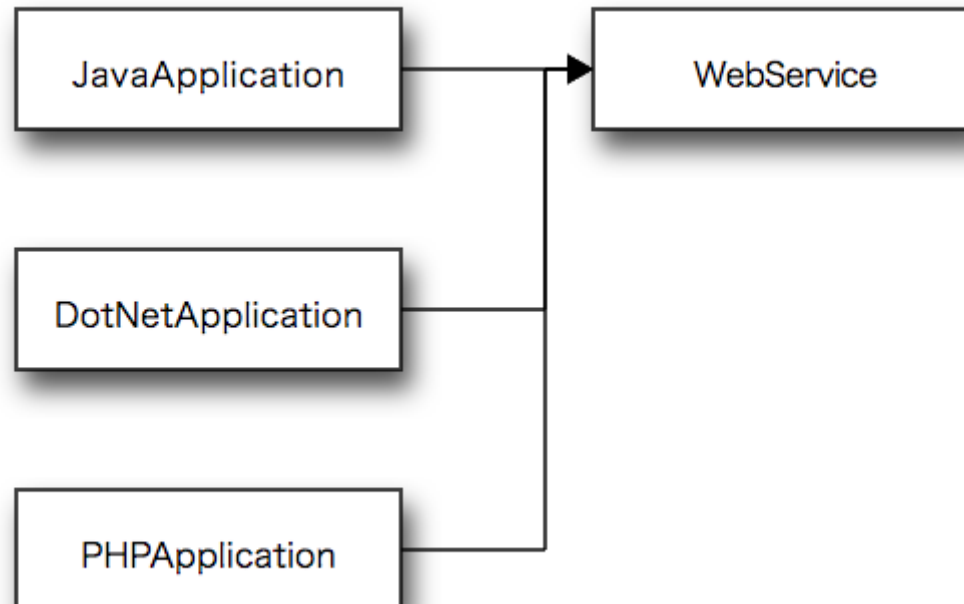
HOW?

*How does data exchange between
applications take place?*





*How can we make web services
platform independent?*



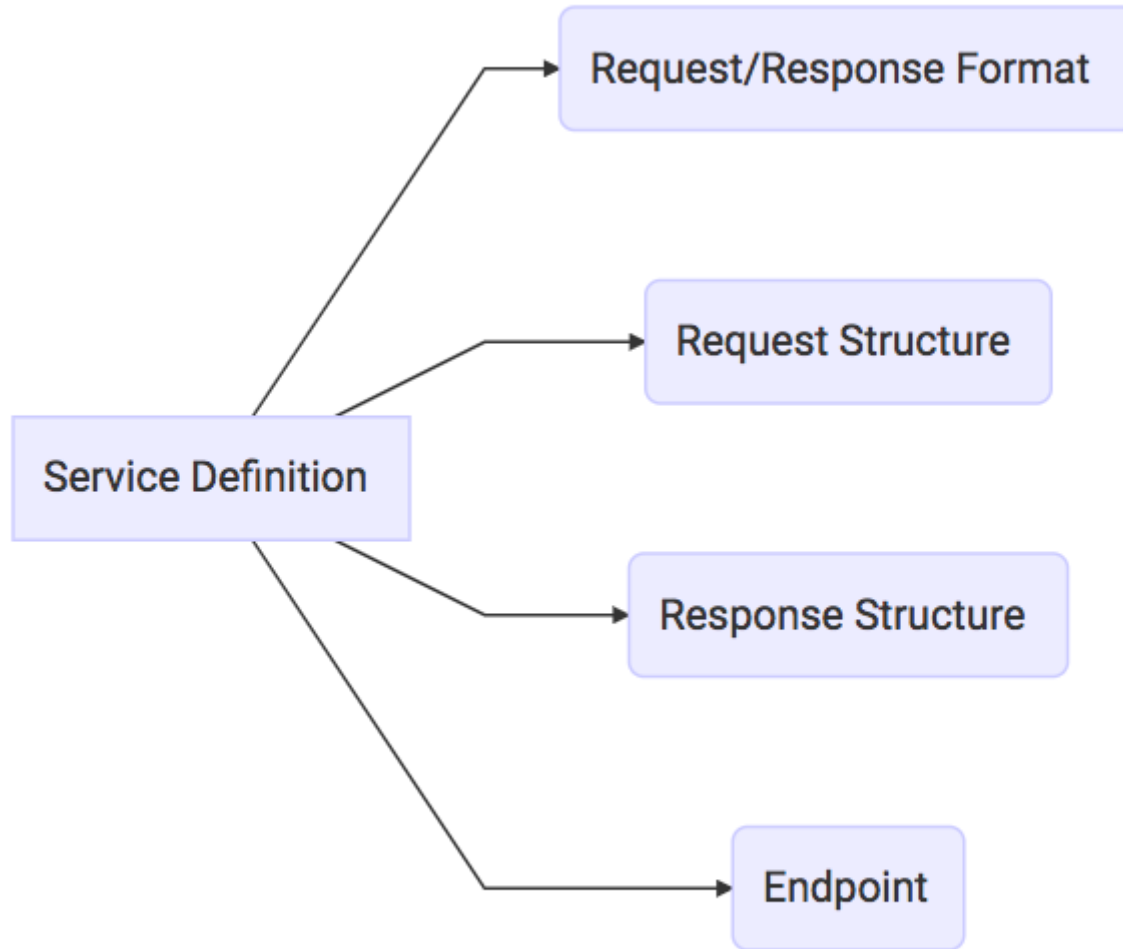
XML

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

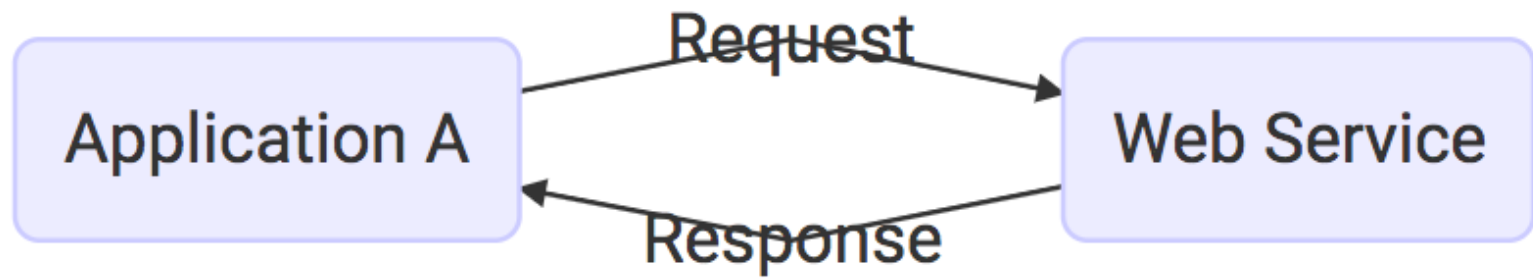
JSON

```
[
  {
    "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

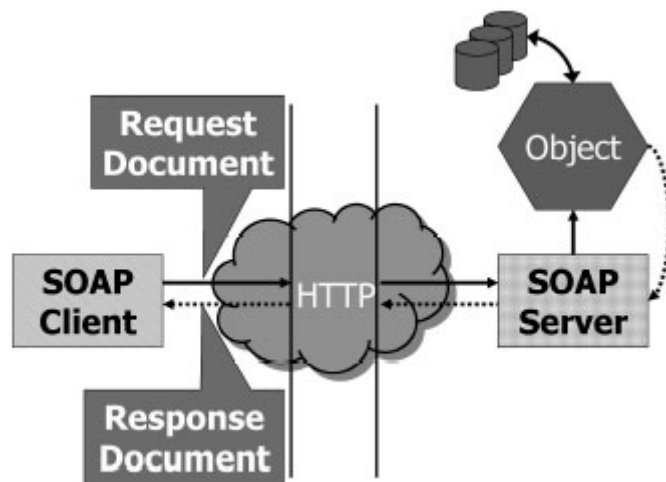
- Request and Response
- Message Exchange Format
 - XML and JSON

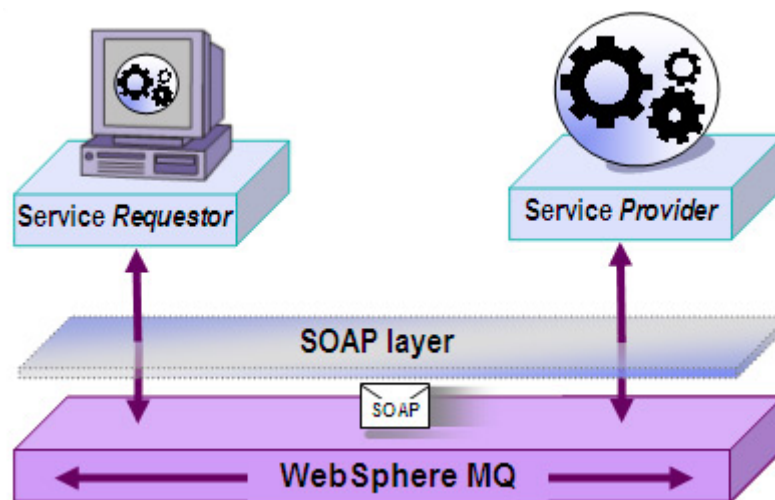
KEY TERMINOLOGY

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

KEY TERMINOLOGY

- Transport
 - HTTP and MQ



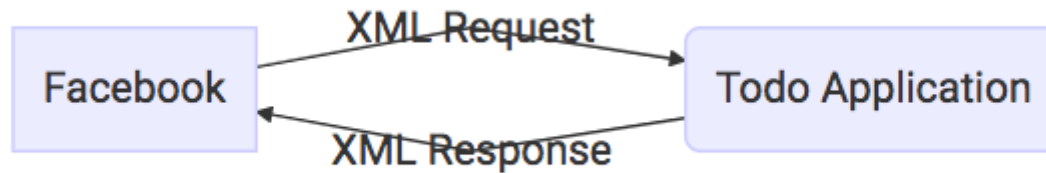


WEB SERVICE GROUPS

- SOAP-based
- REST-styled

*SOAP and REST are not really
comparable.*

SOAP?



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

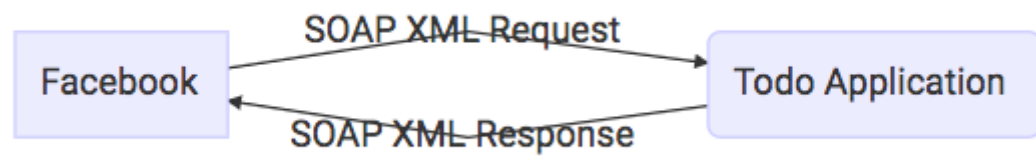
SOAP-ENV: Envelope



The diagram illustrates the structure of a SOAP envelope. It consists of a large light blue rounded rectangle with a black border. Inside this rectangle, at the top, is the text 'SOAP-ENV: Envelope'. Below this text are two smaller rounded rectangles, also with black borders. The first is a yellow rectangle containing the text 'SOAP-ENV: Header'. The second is a larger yellow rectangle containing the text 'SOAP-ENV: Body'.

SOAP-ENV: Header

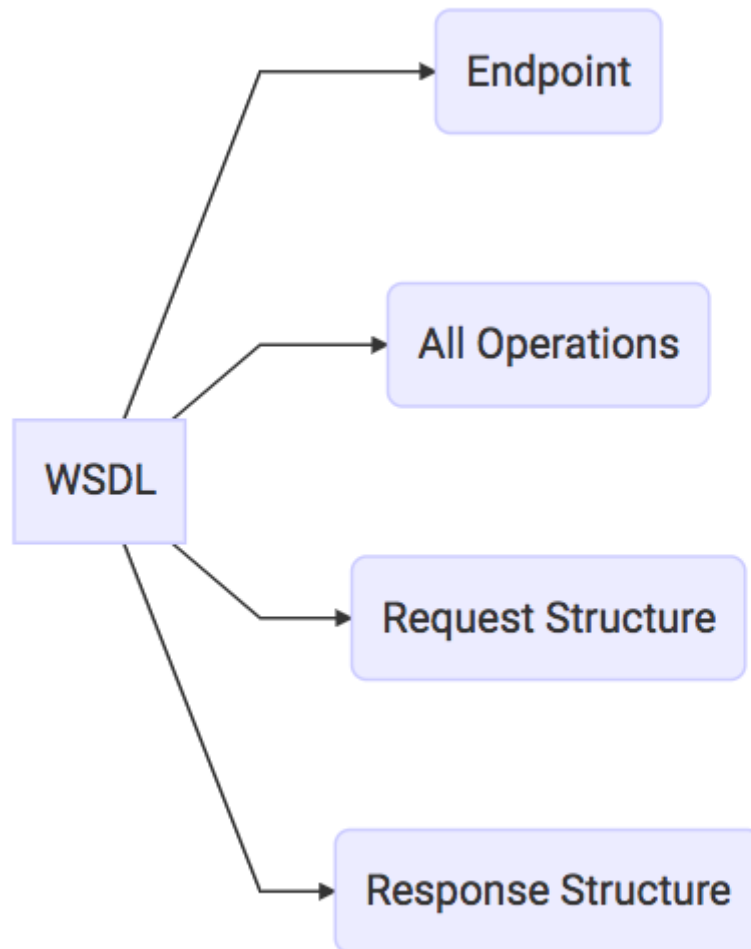
SOAP-ENV: Body

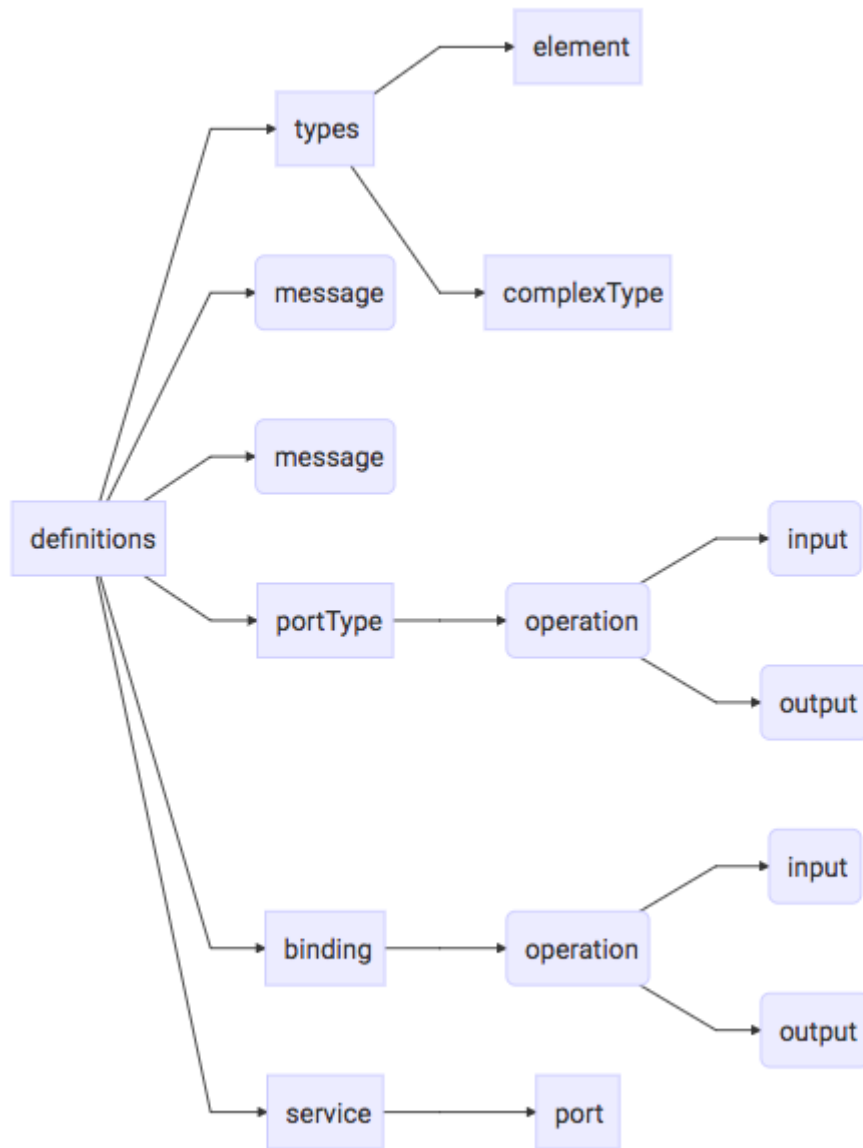


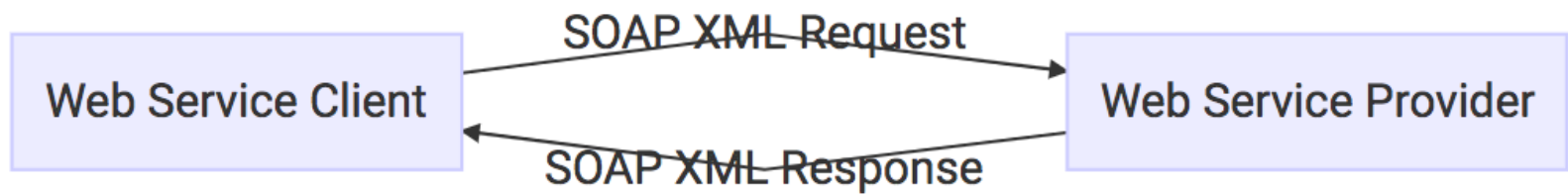
```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:getCourseDetailsResponse xmlns:ns2="http://in28mi
      <ns2:course>
        <ns2:id>Course1</ns2:id>
        <ns2:name>Spring</ns2:name>
        <ns2:description>10 Steps</ns2:description>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

SOAP

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







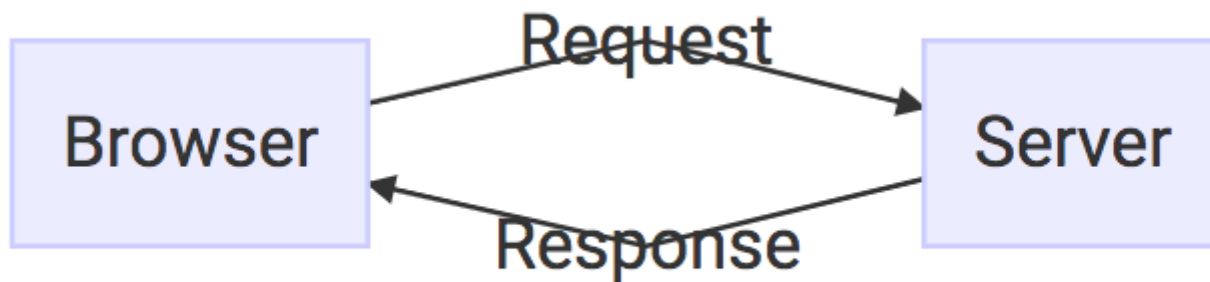
REST

REpresentational State Transfer

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

MAKE BEST USE OF HTTP

| | |
|---------------------------------------|--------------------------------|
| REST(REpresentational State Transfer) | |
| HTTP | |
| HTTP Methods (GET, PUT, POST..) | HTTP Status Codes (200, 404..) |



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

EXAMPLE

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

REST

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

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>
```

REQUEST METHODS

- GET
- POST
- PUT
- DELETE

RESPONSE STATUS

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

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

DEFINE ORGANIZATIONAL STANDARDS

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