Web Services

Web Applications and Services

Spring Term

Naercio Magaia

UNIVERSITY OF SUSSEX

Contents

UNIVERSITY OF SUSSEX

What is Web Services?

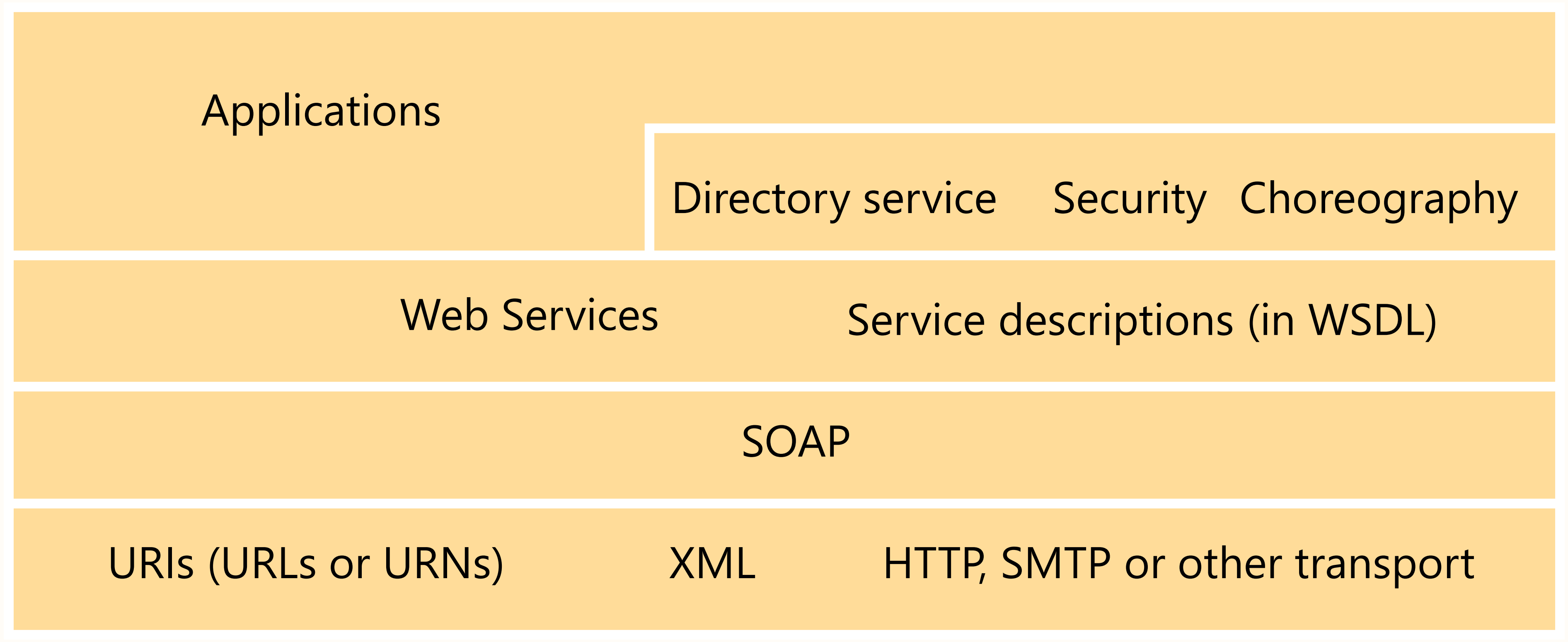
- A Web Service (WS) is a way of utilising services on remote machines
- It is similar to RPC, but with a different Communication Protocol or IDL
- On top of HTTP
- Suited for machine-to-machine communication
- Most popular web sites provide WS to access functionality
- Can be combined with other WS (e.g., aggregator sites)

UNIVERSITY OF SUSSEX

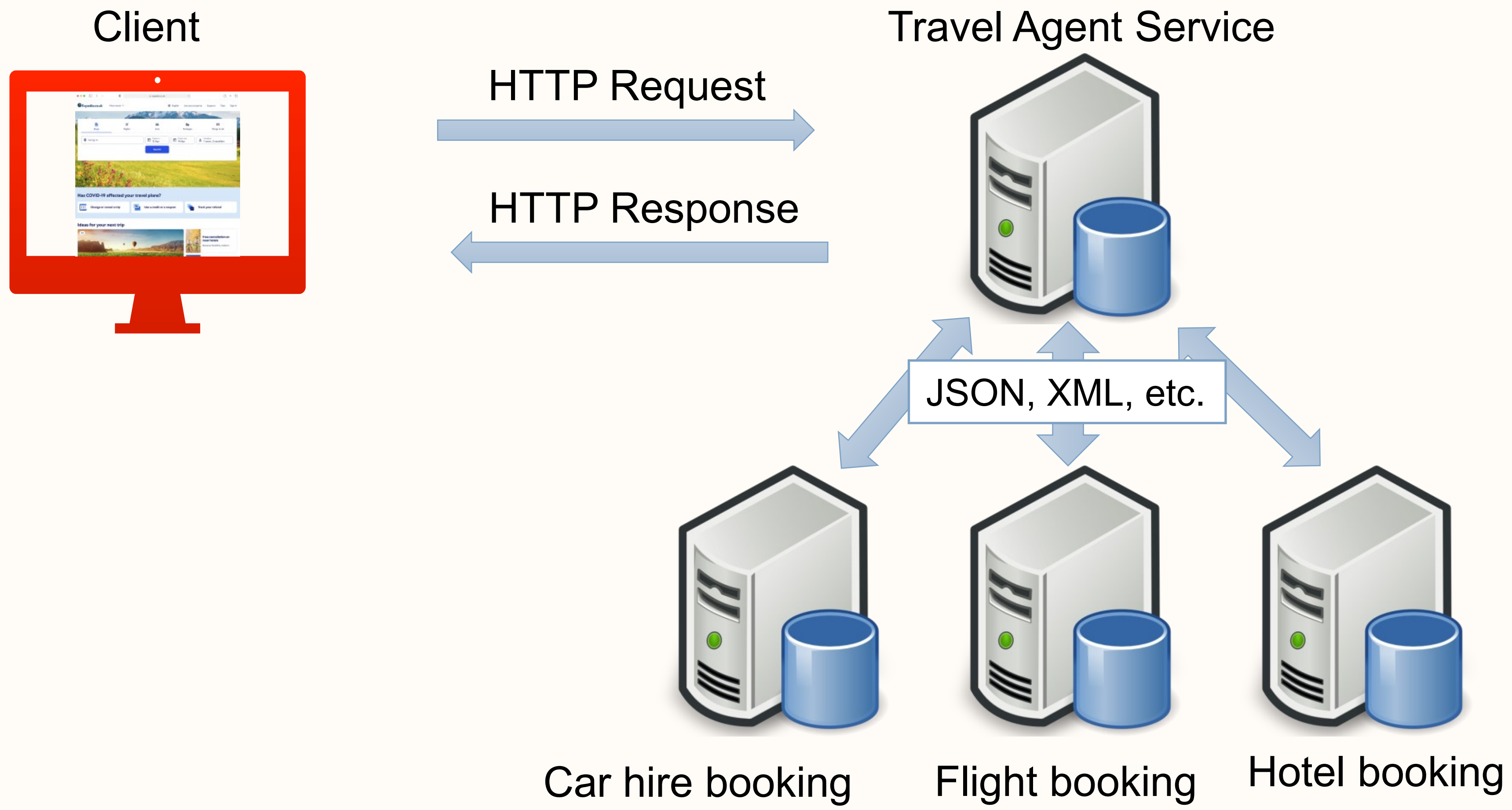
Web Services' usage

- Reusable application-components
  - For example, currency conversion, weather reports, language translation services
- Connect existing software
  - For example, exchange data between different applications and different platforms.
- Access functionality
  - For example, Google web service
- Any application can have a Web Service component

Web Services Infrastructure and Components



The ‘travel agent service’



UNIVERSITY OF SUSSEX

Simple Object Access Protocol (SOAP)

- Defines a scheme for using XML to represent the contents of request and reply messages and for communicating documents
- It specifies
  - How XML is to be used to represent the contents of individual messages
  - Rules as to how recipients of messages should process the XML elements
- A SOAP message is carried in an envelope, which contains
  - an optional header
  - a body

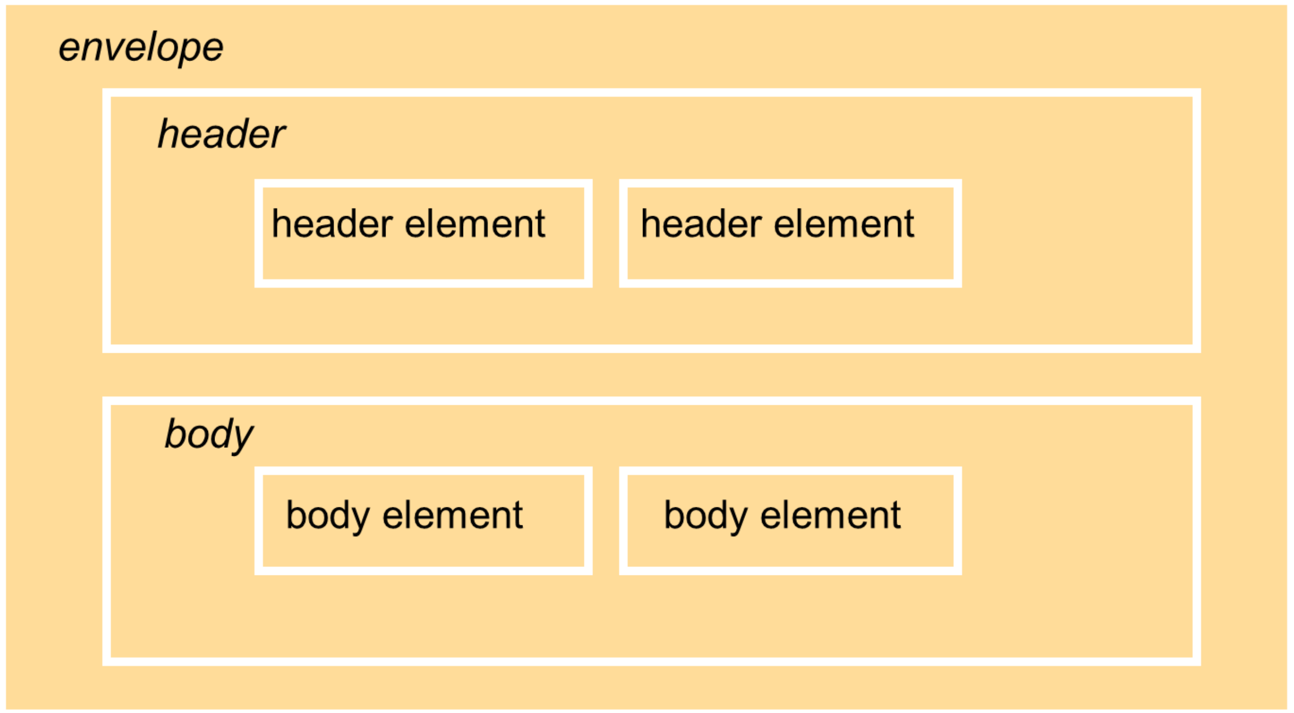
US
UNIVERSITY
OF SUSSEX

SOAP

- It is used for enterprise-level web services that require high security and are complex transactions
- Examples of SOAP APIs
  - financial services,
  - payment gateways,
  - identity management,
  - CRM (Customer Relationship Management),
  - telecommunication services
- PayPal public API

SOAP Message

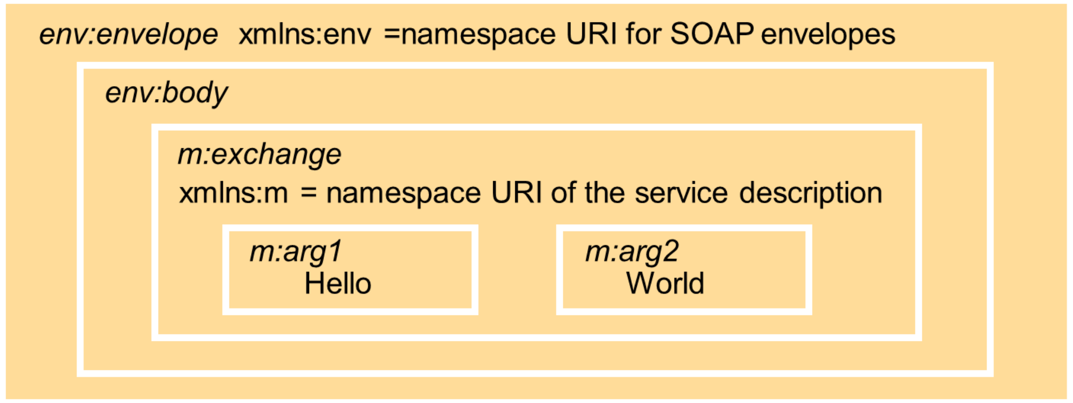
- SOAP message in an envelope



UNIVERSITY OF SUSSEX

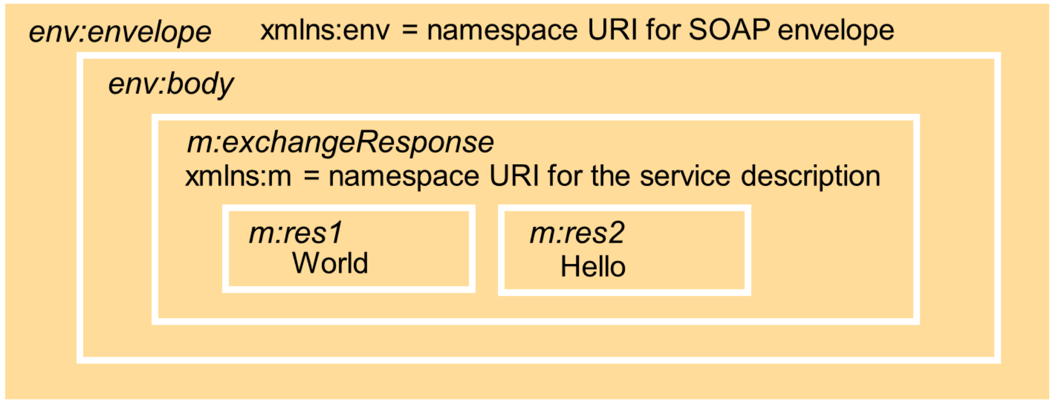
SOAP Message

- A simple request without headers



env:envelope xmlns:env =namespace URI for SOAP envelopes
env:body
m:exchange
xmlns:m = namespace URI of the service description
m:arg1
Hello
m:arg2
World

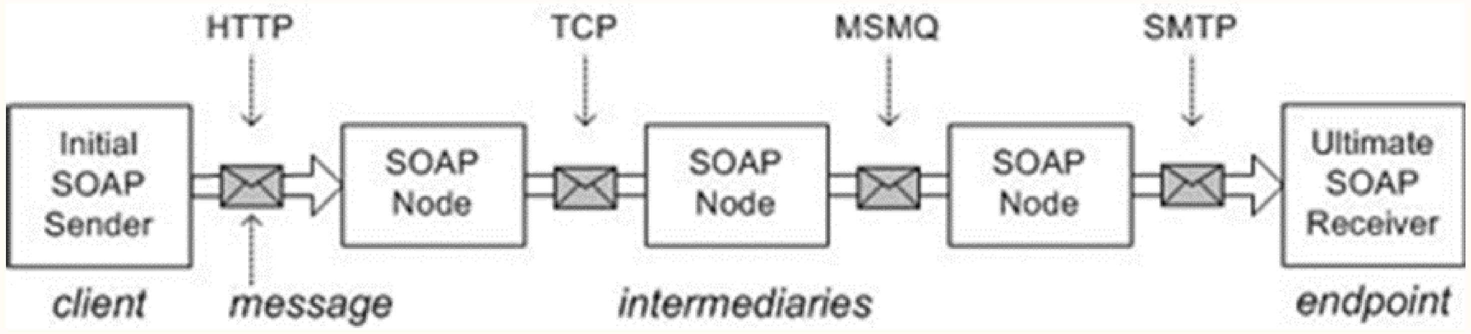
- A simple reply without headers



env:envelope xmlns:env = namespace URI for SOAP envelope
env:body
m:exchangeResponse
xmlns:m = namespace URI for the service description
m:res1
World
m:res2
Hello

SOAP Headers

- Have been designed in anticipation of participation of other SOAP processing nodes, i.e., SOAP intermediaries
  - along a message's path from an initial SOAP sender to an ultimate SOAP receiver
- A SOAP message travels along the message path from a sender to a receiver
- All SOAP messages start with an initial sender, which creates the SOAP message, and end with an ultimate receiver



SOAP Headers

- The *mustUnderstand* attribute means that any node (i.e., computer) processing the SOAP message must understand the given header block
- The *relay* attribute determines if a header block is allowed to be relayed if not processed
- The *encodingStyle* global attribute can be used to indicate the serialization rules used in a SOAP message
- Using the *role* attribute, header blocks (elements) can be targeted at nodes acting in specific roles (e.g., predefined roles are *none*, *next*, *ultimateReceiver*)

```
<env:Header>
  <jj:maxRelayTime value="10000" xmlns:jj="http://mysite.com/myschema"
       role="http://www.w3.org/2003/05/soap-envelope/role/next"
       relay="true"
       mustUnderstant = "false"
   />
</env:Header>
```

UNIVERSITY OF SUSSEX

SOAP Body

- Is the area of the SOAP message, where the application-specific XML data being exchanged in the message is placed
- The `<Body>` element must be present
- It may contain a number of child elements (or empty)
  - Application-specific data (request or response)
  - Fault message, used only when an error occurs

UNIVERSITY OF SUSSEX

SOAP and HTTP

- Using the POST method (common)

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

UNIVERSITY OF SUSSEX

SOAP and HTTP

- Using the GET method (supported)

```
GET /travelcompany.example.org/reservations?code=FT35ZBQ  HTTP/1.1
Host: travelcompany.example.org
Accept: application/soap+xml


HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: 12345

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
...
</env:Envelope>
```

UNIVERSITY OF SUSSEX

SOAP Fault

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn


<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
       <env:Fault>
           <env:Code>
                 <env:Value>env:Sender</env:Value>
                    <env:Subcode>
                         <env:Value>rpc:BadArguments</env:Value>
                    </env:Subcode>
         </env:Code>
         <env:Reason>
                  <env:Text xml:lang="en-US">Processing error</env:Text>
                  <env:Text xml:lang="cs">Chyba zpracování</env:Text>
         </env:Reason>
        </env:Fault>
     </env:Body>
</env:Envelope>
```

UNIVERSITY OF SUSSEX

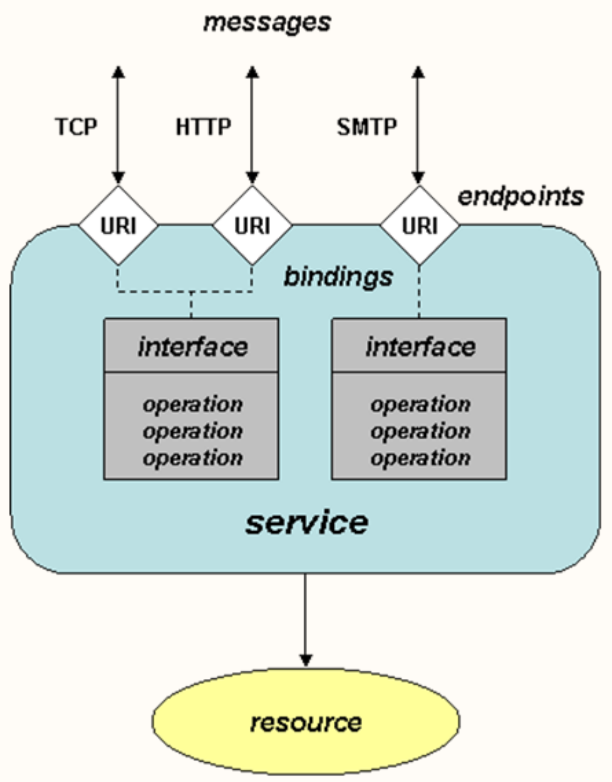
Web Services Description Language (WSDL)

- The IDL for Web Services
- Defined in XML
- Programming Language Independent
- Developers group related operations into interfaces
- Clients must
  - be aware of these groupings
  - know what <u>communication protocol to use</u> for sending messages to the service
  - know the <u>specific mechanics involved</u> in using the given protocol, such as the use of commands, headers, and error codes

UNIVERSITY OF SUSSEX

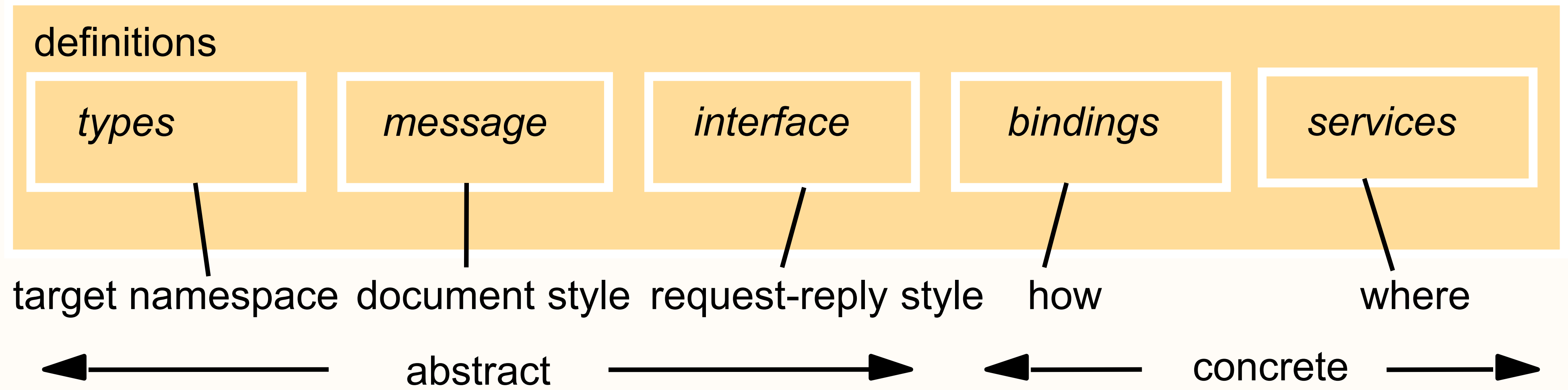
WSDL Bindings

- A binding specifies the concrete details of what goes on the wire
  - How to use an interface with a particular communication protocol?
  - How are abstract messages encoded on the wire?
  - What is the style of service (document vs. RPC)?
- A service can support multiple bindings for a given interface
  - Each binding should be accessible at a unique address identified by a URI (web service endpoint)



UNIVERSITY OF SUSSEX

Main Elements in a WSDL Description



REpresentational State Transfer (REST)

- Focus on resources not on operations and actions
- Intended to return to the principles underpinning the design of the WWW
- Every resource is uniquely addressable
- Stateless and cacheable
- Maps CRUD (Create, Read, Update, Delete) actions to HTTP methods
  - Create → POST
  - Read → GET
  - Update → PUT
  - Delete → DELETE

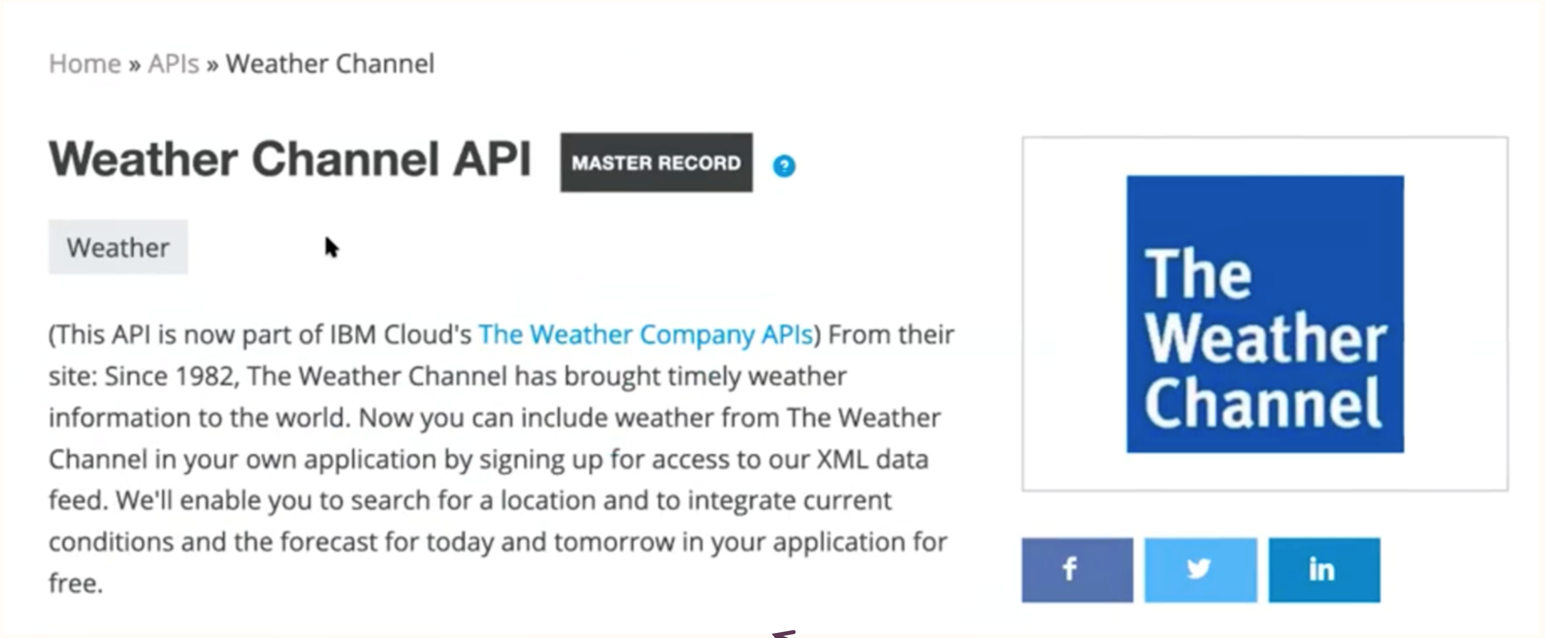
UNIVERSITY OF SUSSEX

REST API Examples

- Some third-party services
  - GitHub REST API
    - https://docs.github.com/en/rest
  - Google Maps/Translate APIs
    - https://developers.google.com/maps/documentation/directions/web-service-best-practices
  - flickr: the App Garden
    - https://www.flickr.com/services/api/
  - Amazon S3 REST API
    - https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html
  - The X REST API
    - https://docs.x.com/x-api/introduction

UNIVERSITY OF SUSSEX

REST API Examples

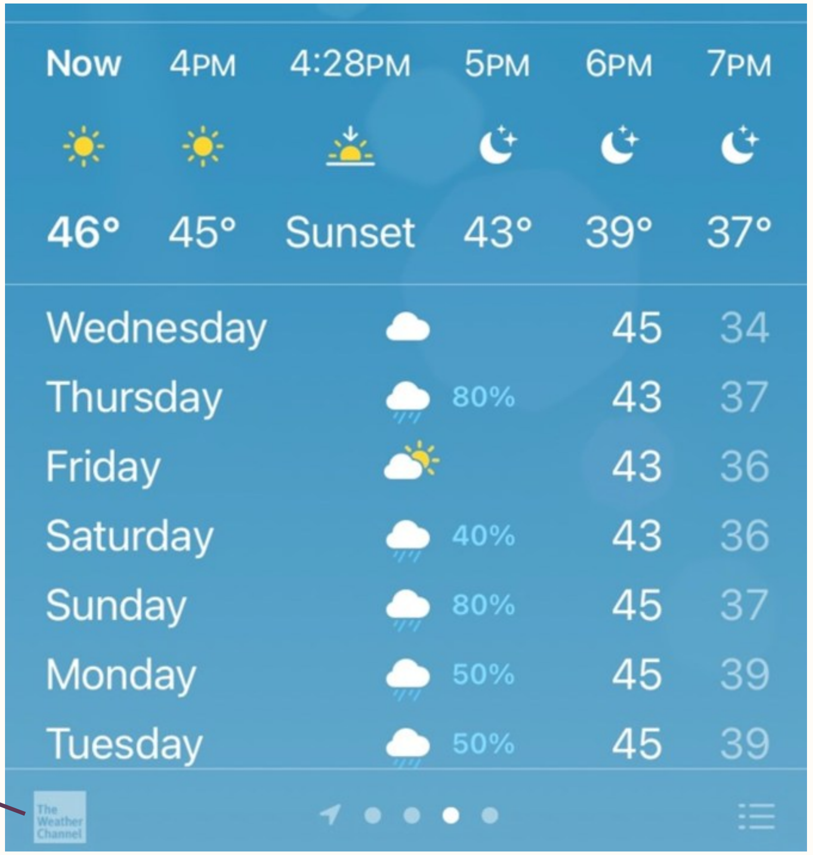


Home » APIs » Weather Channel

Weather Channel API MASTER RECORD

Weather

(This API is now part of IBM Cloud's The Weather Company APIs) From their site: Since 1982, The Weather Channel has brought timely weather information to the world. Now you can include weather from The Weather Channel in your own application by signing up for access to our XML data feed. We'll enable you to search for a location and to integrate current conditions and the forecast for today and tomorrow in your application for free.



Now	4PM	4:28PM	5PM	6PM	7PM
46°	45°	Sunset	43°	39°	37°

Wednesday		45	34
Thursday	80%	43	37
Friday		43	36
Saturday	40%	43	36
Sunday	80%	45	37
Monday	50%	45	39
Tuesday	50%	45	39

The Web as a RESTful service

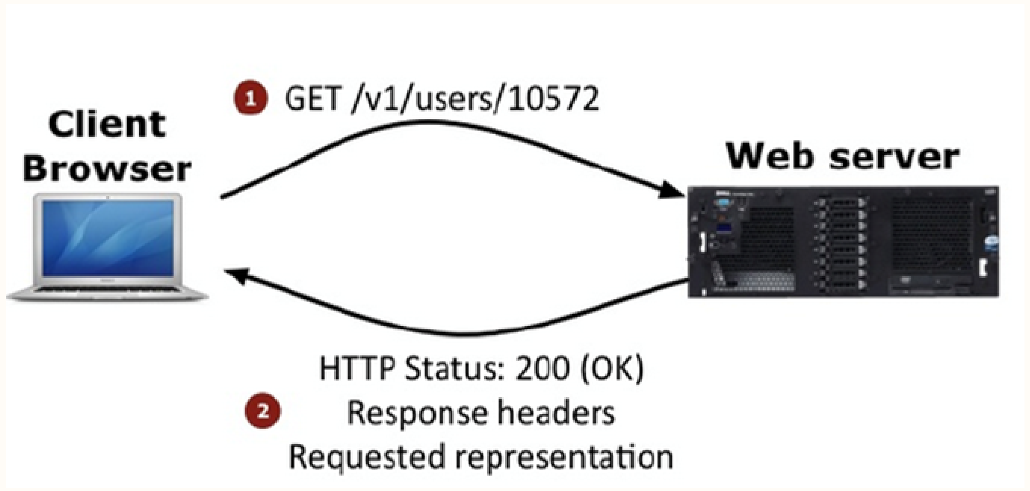
- URIs address resources
- HTTP status codes to report the outcome of a REST operation
- HTTP headers play an active role
- GET is used to access resources that are located at the specified URI on the server.
  - They can cache GET requests and send parameters in the RESTful API request to instruct the server to filter data before sending.

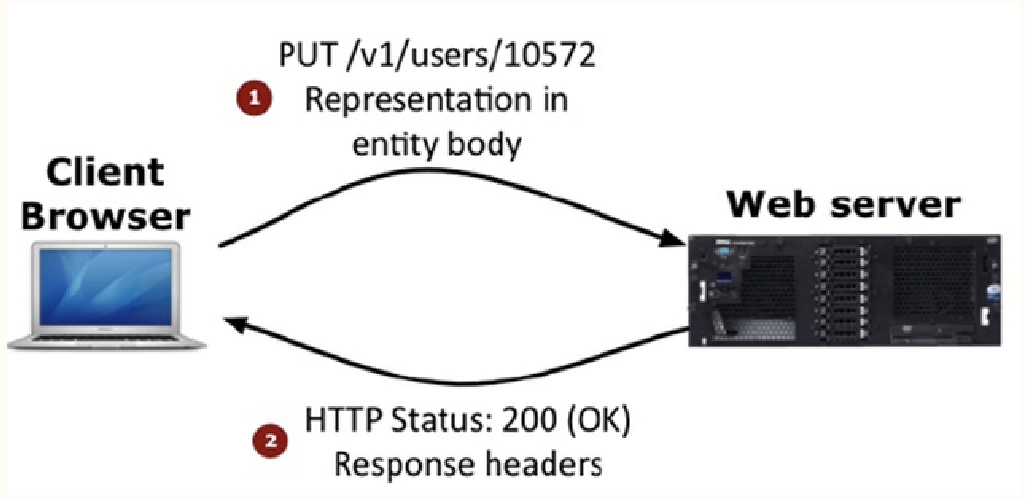
The Web as a RESTful service

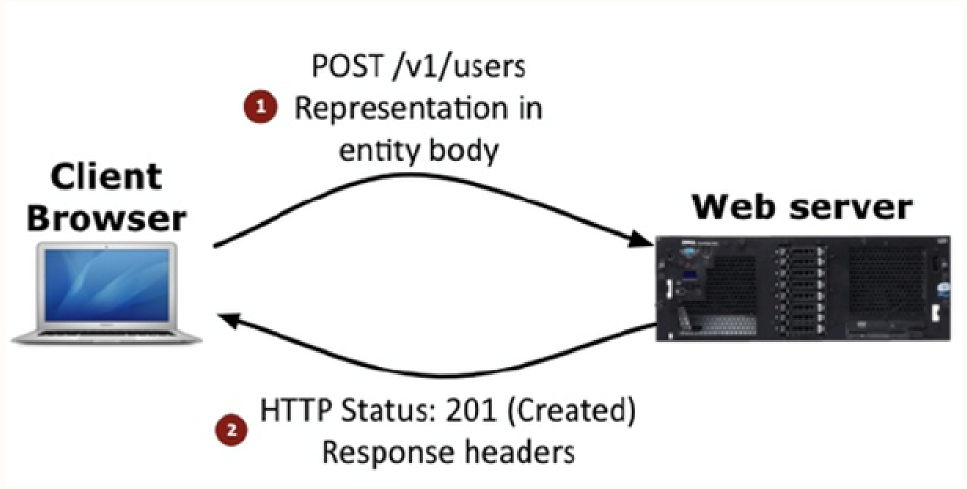
- POST is used to send data to the server.
  - Resource is included with the request.
  - Sending the same POST request multiple times has the side effect of creating the same resource multiple times.
- PUT is used to update existing resources on the server.
  - Unlike POST, sending the same PUT request multiple times in a RESTful web service gives the same result.
- DELETE is used to request the removal of the resource.
  - A DELETE request can change the server state. However, the request fails if the user does not have appropriate authentication.

UNIVERSITY
OF SUSSEX

Example interactions







UNIVERSITY OF SUSSEX

SOAP vs REST

- REST emphasises the web as a set of resources and provides URIs
- SOAP is a protocol which emphasises on operations (e.g., RPC)
- Third-party services (e.g., Amazon, Google, Yahoo) provide multiple APIs
- Use the Web Service approach most suitable for the task at hand

SOAP vs REST

- Protocol
- Allows only XML data format
- SOAP messages consume more bandwidth
- Always makes use of the HTTP POST method
- Function-driven (operations, RPC-like)
- Enterprise apps, high-security apps, financial services, payment gateways
- Poorer performance, more complexity, less flexibility
- It does not use a web caching mechanism

- Architectural style
- Allows different data formats: XML, JSON, TEXT
- Consumes less bandwidth as it utilises HTTP header
- REST takes full advantage of HTTP protocol (i.e., GET, POST, PUT and DELETE)
- Data-driven (resources)
- Public APIs for web services, mobile services, and social networks
- Less security, not suitable for distributed environments
- It uses a web caching mechanism

UNIVERSITY OF SUSSEX

SOAP vs REST

- SOAP example

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
           <soap:Body xmlns:m="http://www.example.org/stock">
                      <m:GetStockPrice>
                                 <m:StockName>IBM</m:StockName>
                      </m:GetStockPrice>
           </soap:Body>
</soap:Envelope>
```

- REST Equivalent

```
GET /stock/IBM HTTP/1.1
Host: www.example.org
Accept: application/xml (and/or application/json…)
```

UNIVERSITY OF SUSSEX

SOAP vs REST

- SOAP example

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
           <soap:Body xmlns:m="http://www.example.org/stock">
                      <m:BuyStock>
                                 <m:StockName>IBM</m:StockName>
                                 <m:Quantity>50</m:Quantity>
                      </m:BuyStock>
           </soap:Body>
</soap:Envelope>
```

- REST Equivalent

```
POST /order HTTP/1.1
Host: www.example.org
Content-Type: application/xml; charset=utf-8

<?xml version="1.0"?>
<order>
          <StockName>IBM</StockName>
          <Quantity>50</Quantity>
</order>
```

UNIVERSITY OF SUSSEX

Caching and REST

- Most resources do not change often
- Caching using HTTP and REST
- Conditional GET

UNIVERSITY OF SUSSEX

Django REST framework

- Is a powerful and flexible toolkit for building Web APIs
- Serialization supports both ORM and non-ORM data sources
- Authentication policies, including packages for OAuth1a and OAuth2
- To enable the REST framework, first install it

```
pip install django_rest_framework
```

- Update the settings.py file

```
INSTALLED_APPS = [
        ...
        'rest_framework',
]
```

Next Lecture …

- ✓ Introduction
- ✓ HTTP, Caching, and CDNs
- ✓ Views
- ✓ Templates
- ✓ Forms
- ✓ Models
- ✓ Security
- ✓ Transactions
- ✓ Remote Procedure Call
- ✓ Web Services
- ➢ **Time**
- Elections and Group Communication
- Coordination and Agreement

UNIVERSITY OF SUSSEX