

CS603 – Web Engineering

PREPARED BY: DR. REEMA PATEL

Web Service

- A web service is a network accessible interface to application functionality, built using standard Internet technologies.
- Clients of web services do **NOT** need to know how it is implemented.
- A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML.
- Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by internet protocols.

Web Services

- Web services can convert your application into a web-application which can publish its function or message to the rest of the world.
- The basic web services platform is XML + HTTP.
- A application which run on web (Internet or Intranet) and provides generic services
- The services provided are through the web and in standardized format which makes It generic and independent on the platform or the protocol on which web services are requested.

Web Service

- Web services are open standard (XML, SOAP, HTTP etc.) based web applications that interact with other web applications for the purpose of exchanging data
- There are two types of web services:
 - SOAP (JAX-WS, Java API for XML services)
 - REST (JAX-RS, JAVA API for RESTful web services)

Web Service

- Web service is, any service that
 - Is available over the Internet or private (intranet) networks
 - Uses a standardized XML messaging system
 - Is not tied to any one operating system or programming language
 - Is self-describing via a common XML grammar
 - Is discoverable via a simple find mechanism

Why web services?

- **Exposing the existing function on to network:**
 - A web service is a unit managed code that can be remotely invoked using HTTP,
 - It can be activated using HTTP requests
 - Allows you to expose the functionality of your existing code over the network
 - Once exposed, others can use the functionality of your program

Why web services?

- **Connecting different applications i.e. Interoperability**
 - Allows different applications to talk to each other and share data and services among themselves
 - For example, VB or .NET application can talk to java web services and vice versa
 - Web services used to make the application platform and technology independant

Why web services?

- Standardized Protocol:
 - Used standardized industry standard protocol for the communication
 - All the four layers (Service Transport, XML messaging, Service Description, and Service Discovery Layers) uses the well defined protocol in the web services protocol stack
 - Standardization of protocol stack gives the business many advantages
 - Like wide range of choices,
 - Reduction in the cost due to competition and increase in the quality

Why web services?

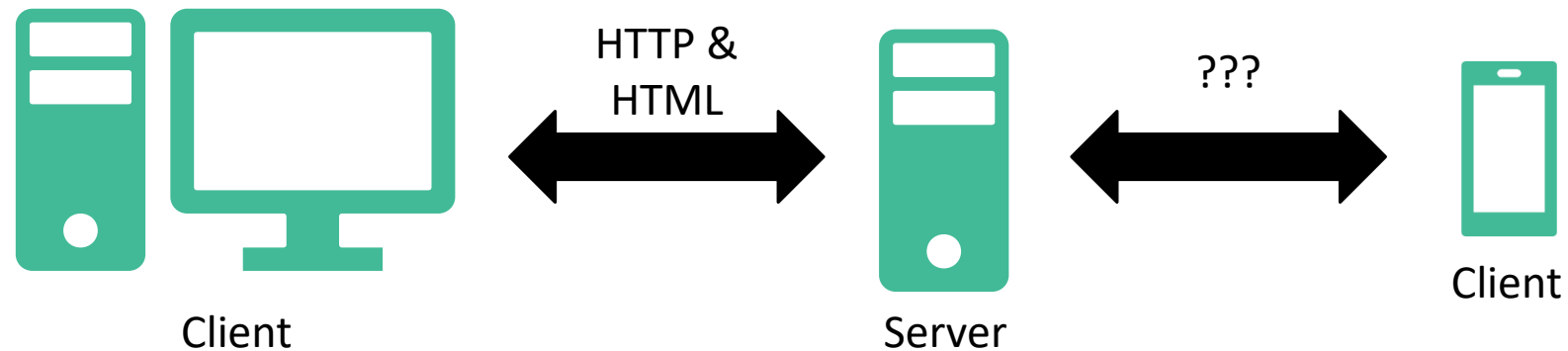
- Low cost of communication
 - Web services uses SOAP over HTTP protocol for communication

Traditional web applications

The interface is built on HTML & HTTP.

- Drawbacks:
 - The client must understand both HTTP and HTML.
 - The entire webpage is replaced with another one.
 - No way to animate transitions between webpages.
 - Same data is usually sent in multiple responses.
 - E.g. HTML code for the layout.

Traditional web applications



- HTTP & HTML can be used, but is not optimal.
 - The GUI on smartphones does not use HTML.
 - E.g. GET /users/3:

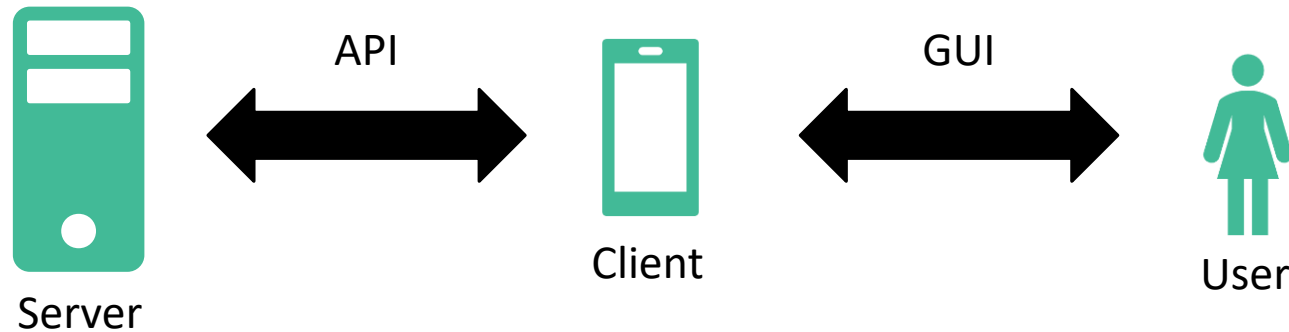
Name Age City

```
<h1>Claire</h1>  
<p>Claire is 24 years old and lives in Boston </p>
```

Application Programming Interface

An API is an interface for Machine ↔ Machine communication.

- An API making use of HTTP is called a *Web API*.

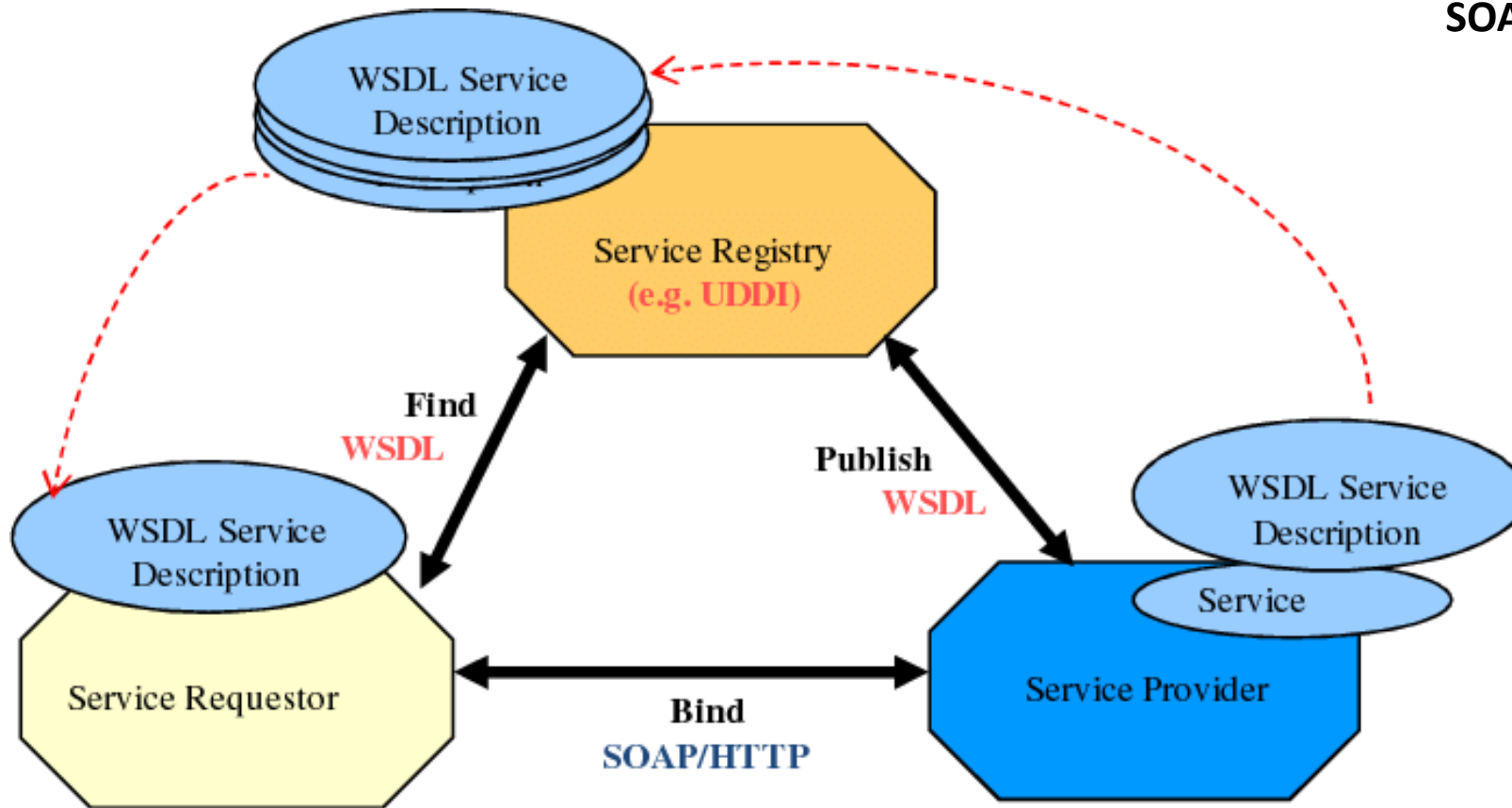


Practical uses of Web Services

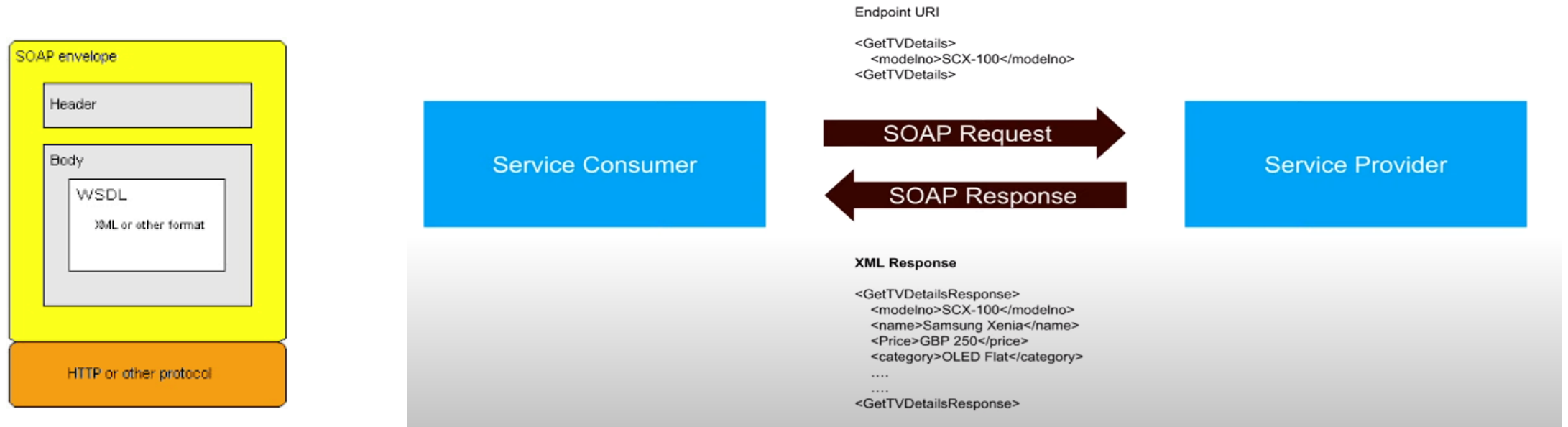
- For Example,
- Retrieve information about TVs from amazon
- Submit an order to amazon
- Update your phone number on Amazon
- Create your own web services to allow these operations on your own website
- Accessing Google map – using web service interacting with google servers

SOAP – Service Oriented Architecture

SOAP - Simple Object Access Protocol



SOAP – Service Oriented Architecture

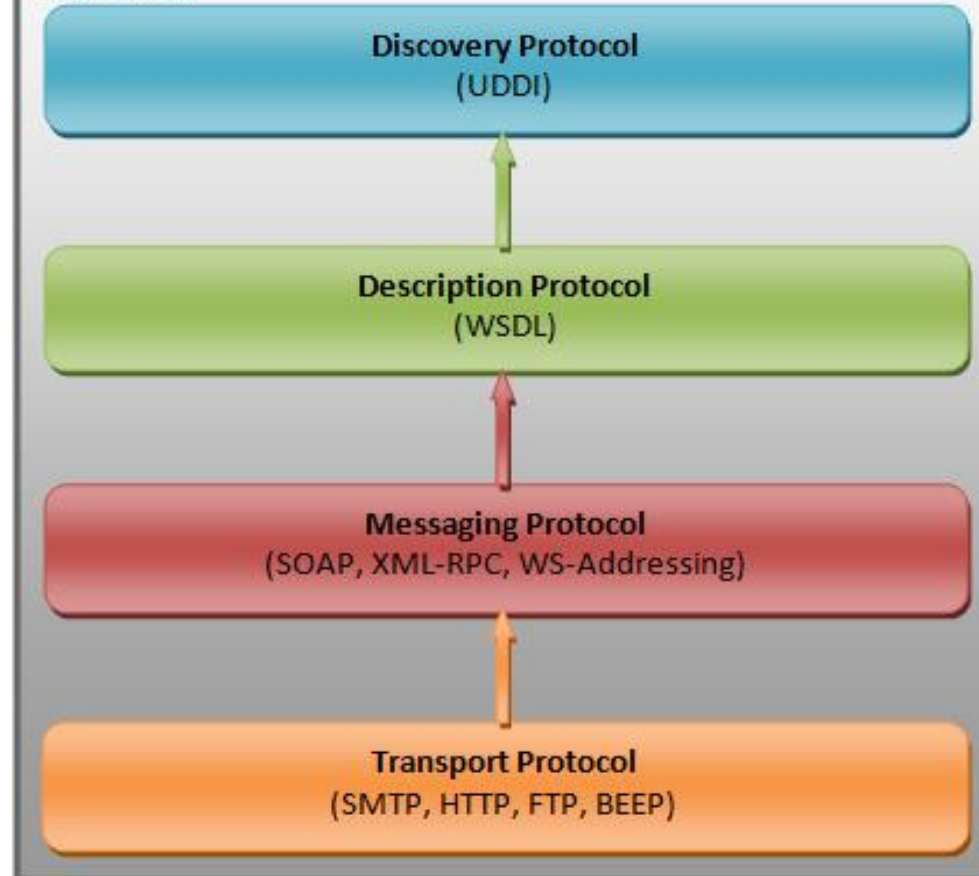


Web Service Roles

- Service Provider: implements the service and makes it available on the Internet
- Service Requestor: Consumer of web service. Requestor utilizes an existing web service by opening a network connection and sending an XML request
- Service Registry: logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones

Web Service Protocol Stack

A **web service protocol stack** is a protocol stack that is used to define, locate, implement, and make **Web services** interact with each other. A Web service protocol stack typically stacks four protocols:



Web Service Protocol Stack

- Service Transport:
 - This layer responsible for transporting messages between applications.
 - This layer includes HTTP, SMTP, FTP, etc. protocols
- XML messaging:
 - This layer encoding messages in a common XML format so that messages can be understood at either end
- Service Description:
 - Responsible for describing the public interface to a specific web service.
 - Via. WSDL
- Service Discovery:
 - Responsible for centralizing services into a common registry, and providing easy publish/find functionality
 - Service discovery handled via Universal Description, Discovery, and Integration (UDDI).

Components of Web Service

- SOAP (Simple Object Access Protocol)
 - Protocol based on XML, used for message transfer
- WSDL (Web Service Description Language)
 - XML file used to describe the web service and how to access them
- UDDI (Universal Description and Discovery Integration)
 - Used to register and search for web service
 - Directory of web service
- XML-RPC
 - For communication
- HTTP

What is SOAP?

- SOAP is an XML based protocol to let application exchange information over HTTP
- SOAP is a communication protocol
- SOAP is a format for sending messages
- SOAP is designed to communicate via Internet
- SOAP is platform and language independent
- SOAP is simple and extensible
- SOAP allows you to get around firewalls
- SOAP is W3C standard

What is SOAP?

- SOAP supports for advanced security mechanisms, for digital signature and encryption
 - ensures it can not be accessed by unauthorized users
- When to use SOAP API?
 - building apps that require a highly secure and reliable method of exchanging data between applications
- application require complex data structure and business logic
- application require advanced security features like digital signature and encryption

What is WSDL?

- WSDL (Web Service Description Language)
- WSDL is based on XML
- WSDL is used to describe web services
- WSDL is used to locate web services
- WSDL is a W3C standard

What is UDDI?

- UDDI (Universal Description and Discovery Integration)
- Is a directory for storing information about web services
- Is a directory of web services interfaces described by WSDL
- UDDI communicates via SOAP

XML-RPC

- Is a simple protocol that uses XML messages to perform RPCs.
- Requests are encoded in XML and sent via HTTP POST
- XML responses are embedded in the body of the HTTP response
- Platform independent
- Allows diverse applications to communicate
- Is the easiest way to get started with web services

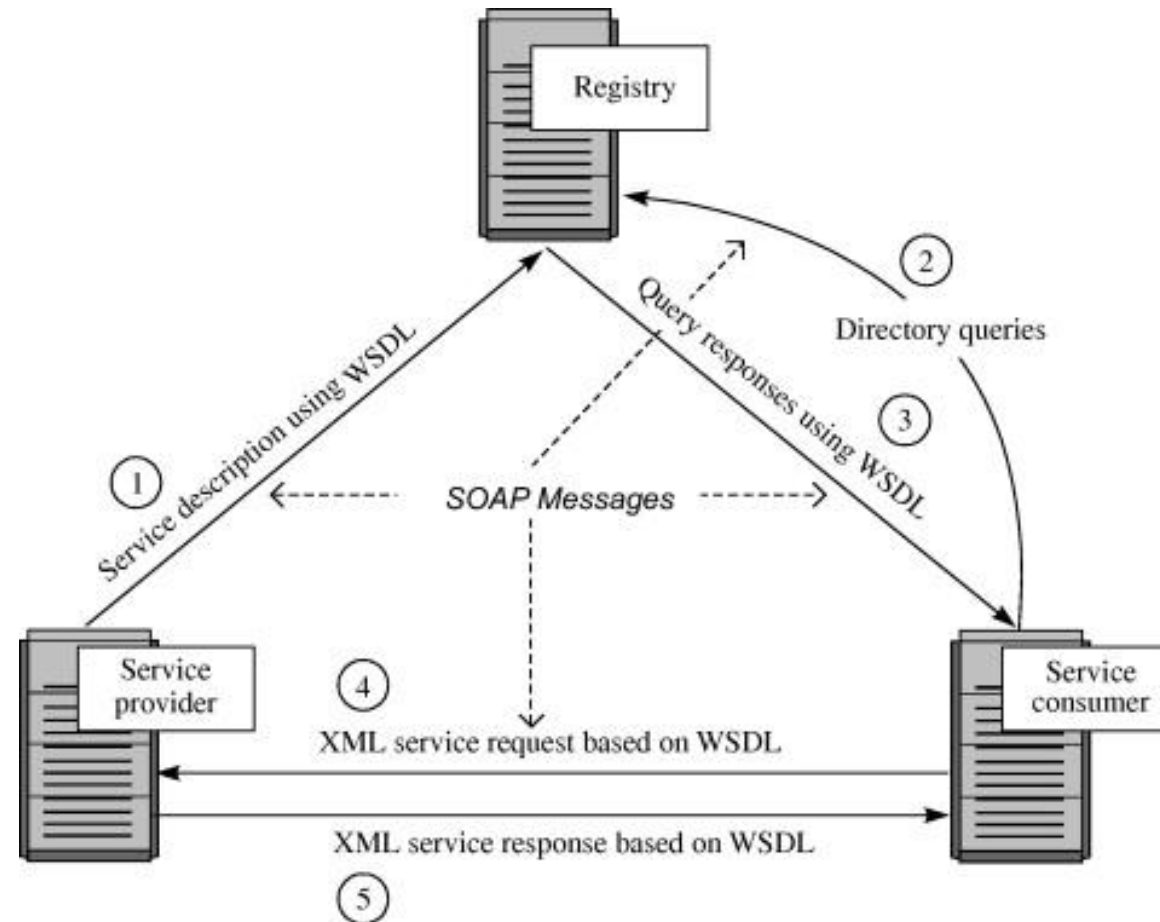
How Web Services Implemented?

- Build and Publish
- Find
- Bind and Invoke

How Web Services Implemented?

- Step 1: Build and Publish
- Build:
 - Create your application
 - Create your WSDL
- Publish:
 - Register your application as a web service onto any registry
 - This process happens on UDDI using separate SOAP request
 - Useful only if your web service should be accessible using Internet

SOAP – Service Oriented Architecture



How Web Services Implemented?

- Step 2: Find:
 - Search in the registry for a web service which provides your needs
 - Obtain the necessary details about service
- Step 3: Bind
 - Use contract file to build the request message
 - Send a request should be sent in the protocol which is required by the service

REST

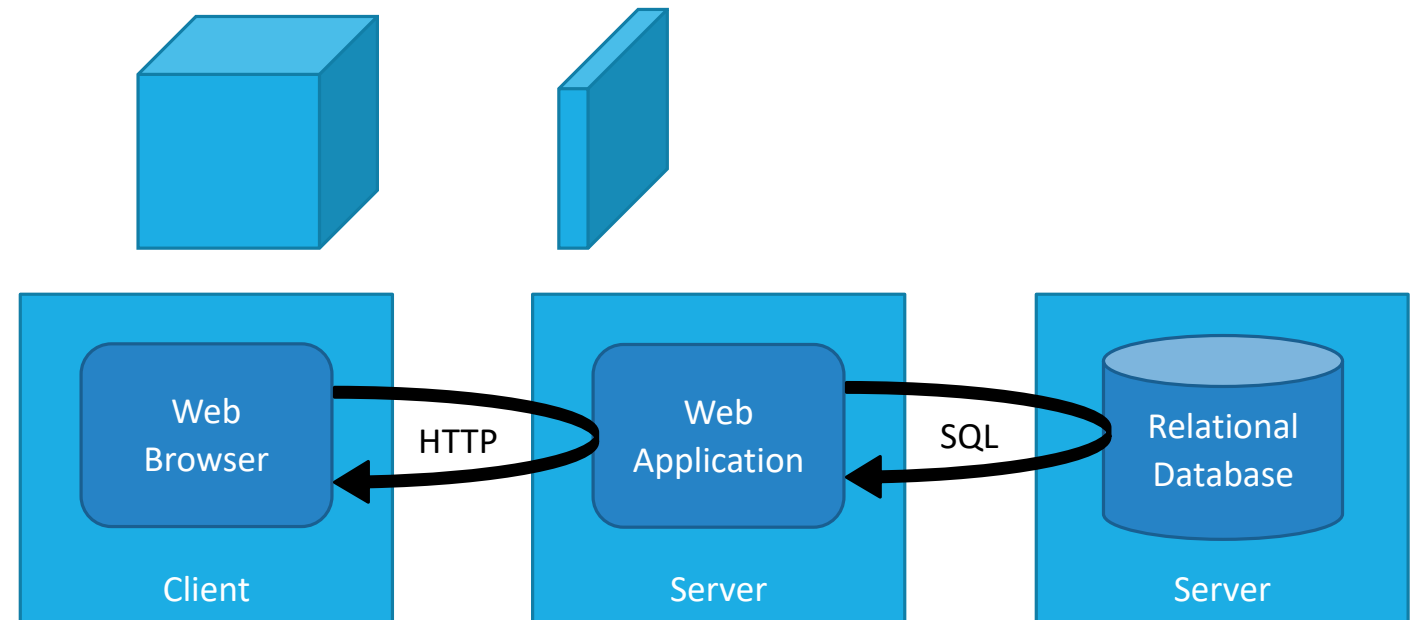
- Representational State Transfer
- Architecture
- Works mostly HTTP
- REST uses HTTP for all 4 CRUD operations – HTTP methods
 - Create (POST)
 - Read (GET)
 - Update (PUT)
 - Delete (DELETE)
- Postcard
- Requires less bandwidth
- REST permits not only XML and file types like JSON can be used
- URI exposes the business logic

REST

- REST architecture uses for building web services (web, mobile, & desktop applications)
- follows restless client-server model – server doesn't store any information about the client's state between requests
 - make REST API scalable and easy to maintain

What is REST?

- An architectural style for distributed hypermedia systems described by Roy Thomas Fielding in his doctoral dissertation 2000.
- Consists of constraints:
 1. Client - Server
 2. Stateless
 3. Cache
 4. Uniform Interface
 5. Layered System
 6. Code-On-Demand



What does REST mean?

- The name "Representational State Transfer" is intended to evoke an image of how a well-designed Web application behaves:
 - a network of web pages (a virtual state-machine),
 - where the user progresses through the application by selecting links (state transitions),
 - resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use.

What does REST mean?



Client

GET /users/2



Server

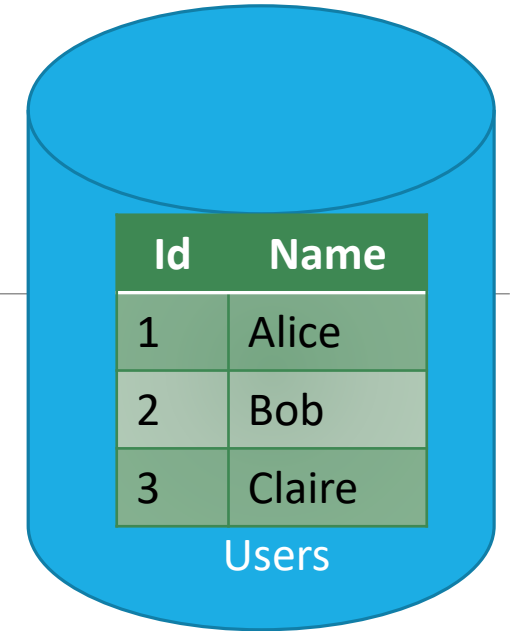
...

{ "id": 2, "name": "Bob" }

Changes state.

{ "id": 2,
"name": "Bob" }

PUT /users/2
{ "id": 2, "name": "Bob" }



Using HTTP as the uniform interface

- Use URIs to identify resources.
- Use HTTP methods to specify operation:
 - Create: POST (or PUT)
 - Retrieve: GET
 - Update: PUT (or PATCH)
 - Delete: DELETE
- Use HTTP headers
Content-Type and Accept
to specify data format for the resources.
- Use HTTP status code to indicate success/failure.

Bad

POST /login

POST /create-book

GET /get-top-10-books

Good

POST /login-sessions

POST /books

GET /top-10-books

REST Example

A server with information about users.

- The GET method is used to retrieve resources.

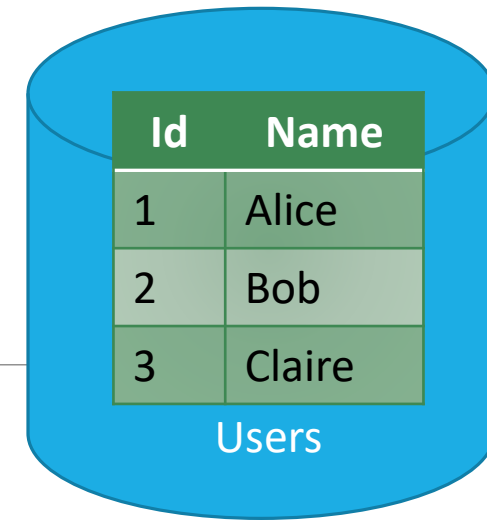
- GET /users
- GET /users/2
- GET /users/pages/1
- GET /users/gender/female
- GET /users/age/18
- GET /users/???
- GET /users/2/name
- GET /users/2/pets

GET /users?page=1

GET /users?gender=female

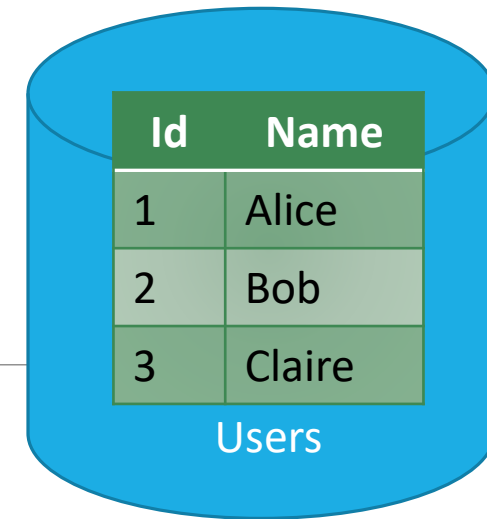
GET /users?age=18

GET /users?gender=female&age=18



REST Example

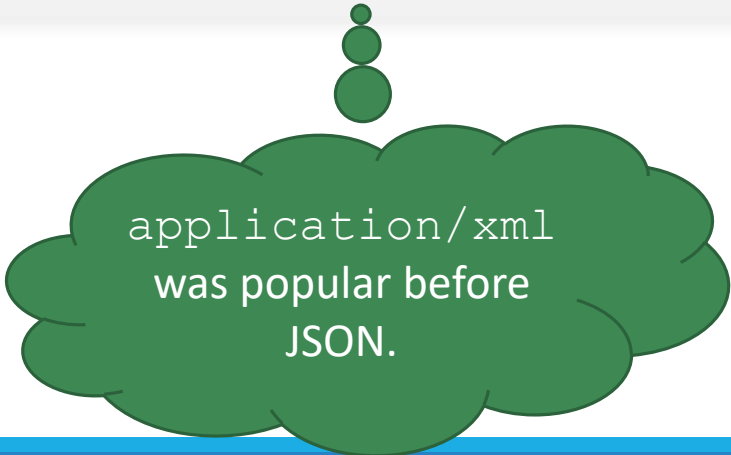
A server with information about users.



Id	Name
1	Alice
2	Bob
3	Claire

- The GET method is used to retrieve resources.
 - Which data format? Specified by the `Accept` header!

```
GET /users HTTP/1.1
Host: the-website.com
Accept: application/json
```



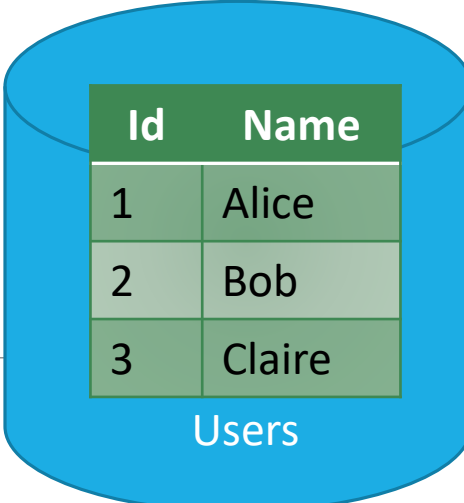
application/xml
was popular before
JSON.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 66
```

```
[
  {"id": 1, "name": "Alice"},
  {"id": 2, "name": "Bob"}
]
```

REST Example

A server with information about users.



Id	Name
1	Alice
2	Bob
3	Claire

Users

- The POST method is used to create resources.
 - Which data format? Specified by the `Accept` and `Content-Type` header!

```
POST /users HTTP/1.1
Host: the-website.com
Accept: application/json
Content-Type: application/xml
Content-Length: 49
```

```
<user>
  <name>Claire</name>
</user>
```

```
HTTP/1.1 201 Created
Location: /users/3
Content-Type: application/json
Content-Length: 28
```

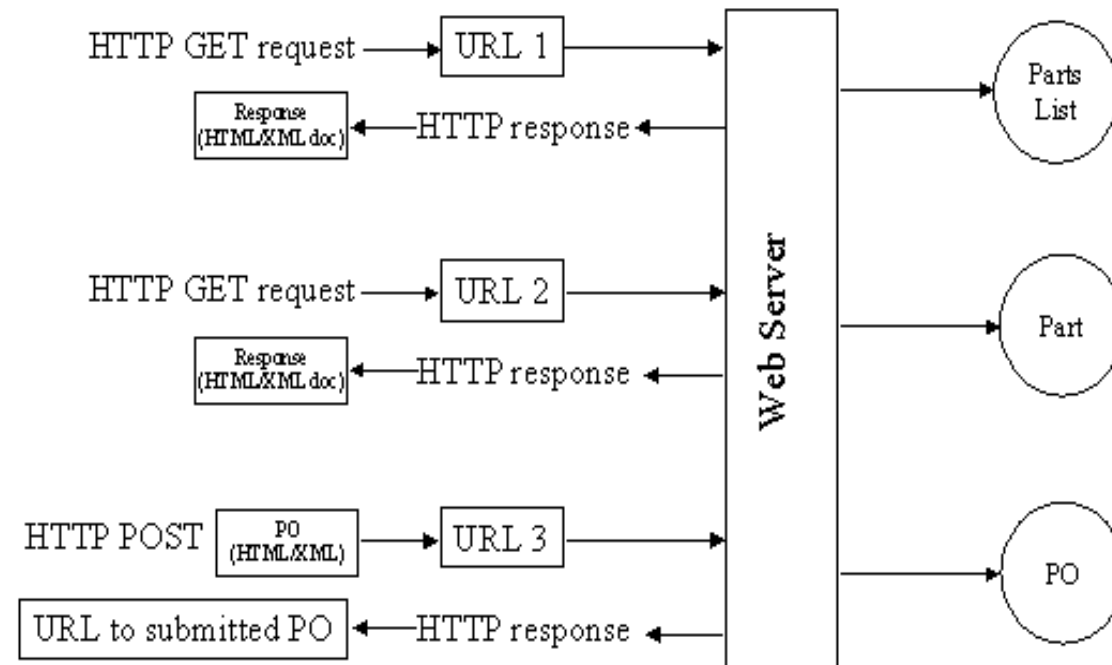
```
{"id": 3, "name": "Claire"}
```

Example 1

Parts Depot Web Services

- Parts Depot, Inc has deployed some web services to enable its customers to:
 - get a list of parts
 - get detailed information about a particular part
 - submit a Purchase Order (PO)

Example 1 - REST way of Implementing the web services



Example 1- Service – Get parts list

The web service makes available a URL to a parts list resource

Client uses : <http://www.parts-depot.com/parts>

Document Client receives :

```
<?xml version="1.0"?>
```

```
<p:Parts xmlns:p="http://www.parts-depot.com" xmlns:xlink="http://www.w3.org/1999/xlink">
```

```
  <Part id="00345" xlink:href="http://www.parts-depot.com/parts/00345"/>
```

```
  <Part id="00346" xlink:href="http://www.parts-depot.com/parts/00346"/>
```

```
  <Part id="00347" xlink:href="http://www.parts-depot.com/parts/00347"/>
```

```
  <Part id="00348" xlink:href="http://www.parts-depot.com/parts/00348"/>
```

```
</p:Parts>
```


Example 1 - Service – Get detailed part data

The web service makes available a URL to each part resource.

Client uses : <http://www.parts-depot.com/parts/00345>

Document Client receives :

```
<?xml version="1.0"?>
```

```
<p:Part xmlns:p="http://www.parts-depot.com" xmlns:xlink="http://www.w3.org/1999/xlink">
```

```
  <Part-ID>00345</Part-ID>
```

```
  <Name>Widget-A</Name>
```

```
  <Description>This part is used within the frap assembly</Description>
```

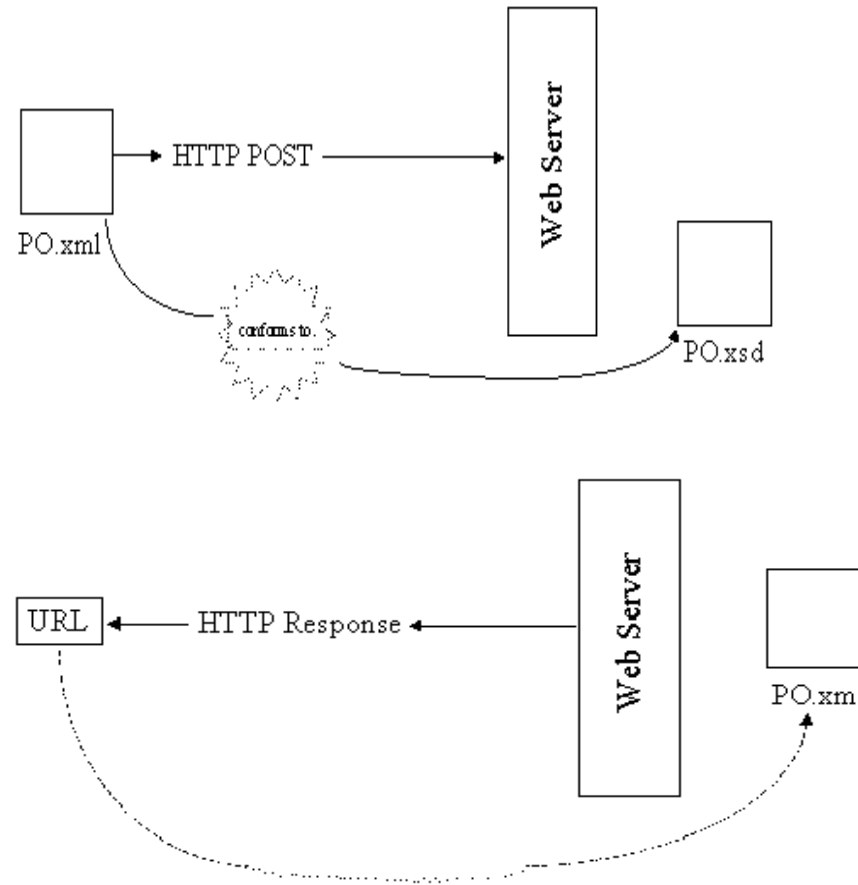
```
  <Specification xlink:href="http://www.parts-depot.com/parts/00345/specification"/> <UnitCost currency="USD">0.10</UnitCost>
```

```
  <Quantity>10</Quantity>
```

```
</p:Part>
```

Example 1- Service – Submit purchase order (PO)

- The web service makes
- available a URL to submit a PO.
- The client creates a PO instance document (PO.xml)
- Submits the PO.xml(HTTP POST)
- PO service responds with a URL to the submitted PO.



Characteristics of a REST based network

- Client-Server: a pull-based interaction style(Client request data from servers as and when needed).
- Stateless: each request from client to server must contain all the information necessary to understand the request, and cannot take advantage of any stored context on the server.
- Cache: to improve network efficiency, responses must be capable of being labeled as cacheable or non-cacheable.
- Uniform interface: all resources are accessed with a generic interface (e.g., HTTP GET, POST, PUT, DELETE).
- Named resources - the system is comprised of resources which are named using a URL.
- Interconnected resource representations - the representations of the resources are interconnected using URLs, thereby enabling a client to progress from one state to another.

Principles of REST web service design

1. Identify all the conceptual entities that we wish to expose as services. (Examples we saw include resources such as : parts list, detailed part data, purchase order)
2. Create a URL to each resource.
3. Categorize our resources according to whether clients can just receive a representation of the resource (using an HTTP GET), or whether clients can modify (add to) the resource using HTTP POST, PUT, and/or DELETE).
4. All resources accessible via HTTP GET should be side-effect free. That is, the resource should just return a representation of the resource. Invoking the resource should not result in modifying the resource.

Principles of REST web service design

5. Put hyperlinks within resource representations to enable clients to drill down for more information, and/or to obtain related information.
6. Design to reveal data gradually. Don't reveal everything in a single response document. Provide hyperlinks to obtain more details.
7. Specify the format of response data using a schema (DTD, W3C Schema, RelaxNG, or Schematron). For those services that require a POST or PUT to it, also provide a schema to specify the format of the response.
8. Describe how our services are to be invoked using either a WSDL document, or simply an HTML document.

When to use REST API?

- Building web services that require a stateless, scalable, & easy-to-maintain architecture
- Apps with CRUD operations (creating, reading, updating, & deleting data)
- For real time communication applications such as chat applications and streaming services

- REST based web services
 - Online shopping
 - Search services
 - Dictionary services

REST Web Service

- RESTful web service is a lightweight, maintainable, and scalable service that is built on the REST architecture.
- RESTful web service, expose API from your application in a secure, uniform, stateless manner to the calling client. The calling client can perform predefined operations using the RESTful service.

- Sample Message:

- ```
{
 "name": "john",
 "age": 30,
 "car": "Tata"
}
```

| GET    | Read or retrieve data           |
|--------|---------------------------------|
| POST   | Add new data                    |
| PUT    | Update data that already exists |
| DELETE | Remove data                     |

## SOAP vs. REST

|                             | SOAP                                                                                                                           | REST                                                                                                                                                                                               |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| What it stands for          | Simple Object Access Protocol                                                                                                  | Representational State Transfer                                                                                                                                                                    |
| Resources                   | XML and HTTP                                                                                                                   | HTTP                                                                                                                                                                                               |
| Skill Level                 | High                                                                                                                           | Low to Medium                                                                                                                                                                                      |
| Data Format                 | Options include XML, JSON, CSV                                                                                                 | Options include JSON, XML, CSV, and other structured formats                                                                                                                                       |
| Design Focus                | Standardization, performance, security, reliability, and transactional support                                                 | Flexibility, interoperability, scalability, simplicity, and statelessness                                                                                                                          |
| Architecture                | SOAP APIs are independent and can work with any transport protocol. This makes them versatile, but also more complex and slow. | REST APIs rely on the underlying transport protocol, usually HTTPS. This means that they can perform better than SOAP APIs, but this can cause challenges with backward compatibility or security. |
| Request and Response format | Requires a standardized structure, including headers and a message body.                                                       | Doesn't require strict structure and usually includes an HTTP method, an endpoint, headers, and a body.                                                                                            |
| Security                    | Provides standards-based security measures                                                                                     | Offers several security measures, such as SSL, OAuth, and HTTP Basic Authentication                                                                                                                |
| Used in                     | Web and non-web applications                                                                                                   | Mostly web applications                                                                                                                                                                            |



| REST API                                                                    | SOAP API                                                                              |
|-----------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| REST Architectural style                                                    | Uses XML as data format                                                               |
| HTTP methods (GET, POST, PUT, DELETE)                                       | Follows specific set of rules and protocols for communication                         |
| Returns data in JSON or XML format only                                     | SOAP specific protocol, such as WSDL, UDDI                                            |
| Stateless client-server model                                               | Suitable for building enterprise level apps – require robust security and reliability |
| Suitable for simple applications – CRUD operations, real time communication |                                                                                       |
|                                                                             |                                                                                       |