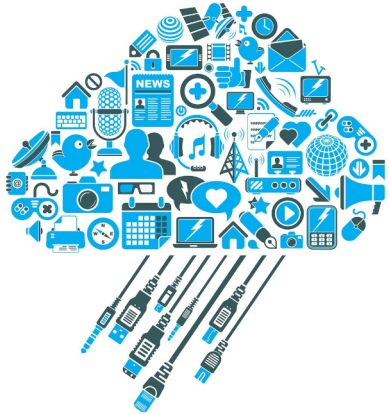


University of CINCINNATI

Concepts of REST
(Representational State Transfer)

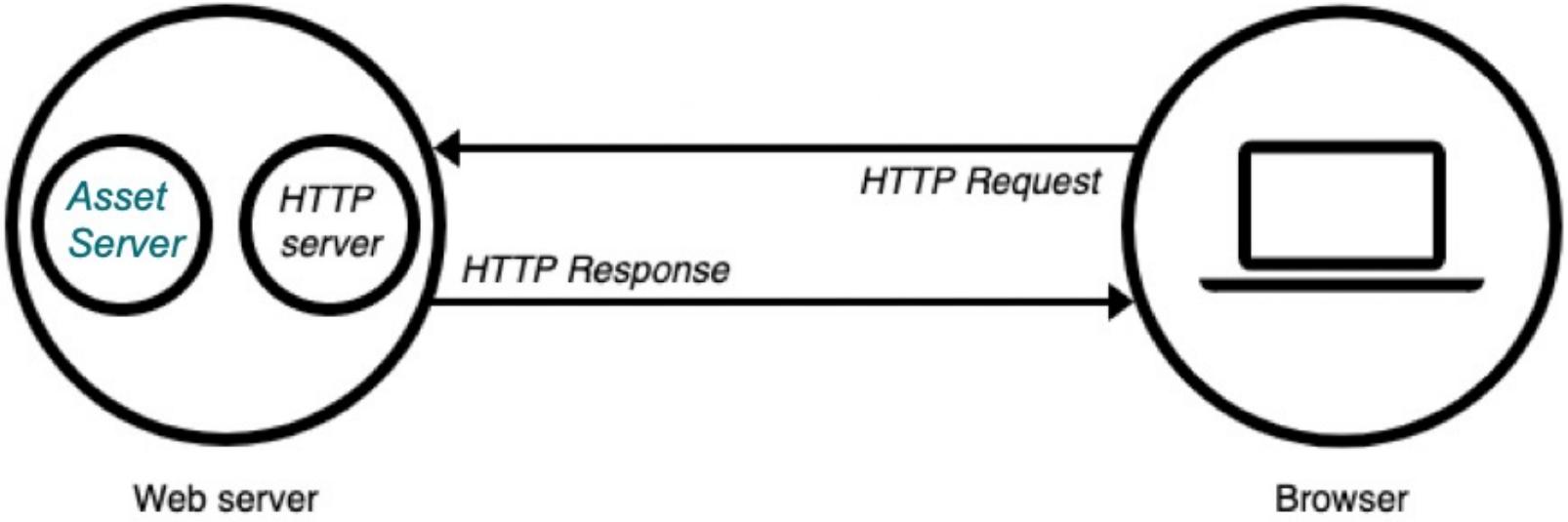
Source : https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Rashmi Kansakar

Asset Server

University of CINCINNATI

- Web server is HW+ SW, used to **host websites**, **store web server contents**, **interchange network requests connected to WEB** and deliver web pages to end users.
- **Web server is used to store the website's files: e.g. HTML documents and their related assets, images, CSS stylesheets, JavaScript files, fonts, and videos**.
- A best practice is to **store them all on a dedicated Web server**, called **ASSET SERVER** or Hosting File Server.
- Static web server and Dynamic web server



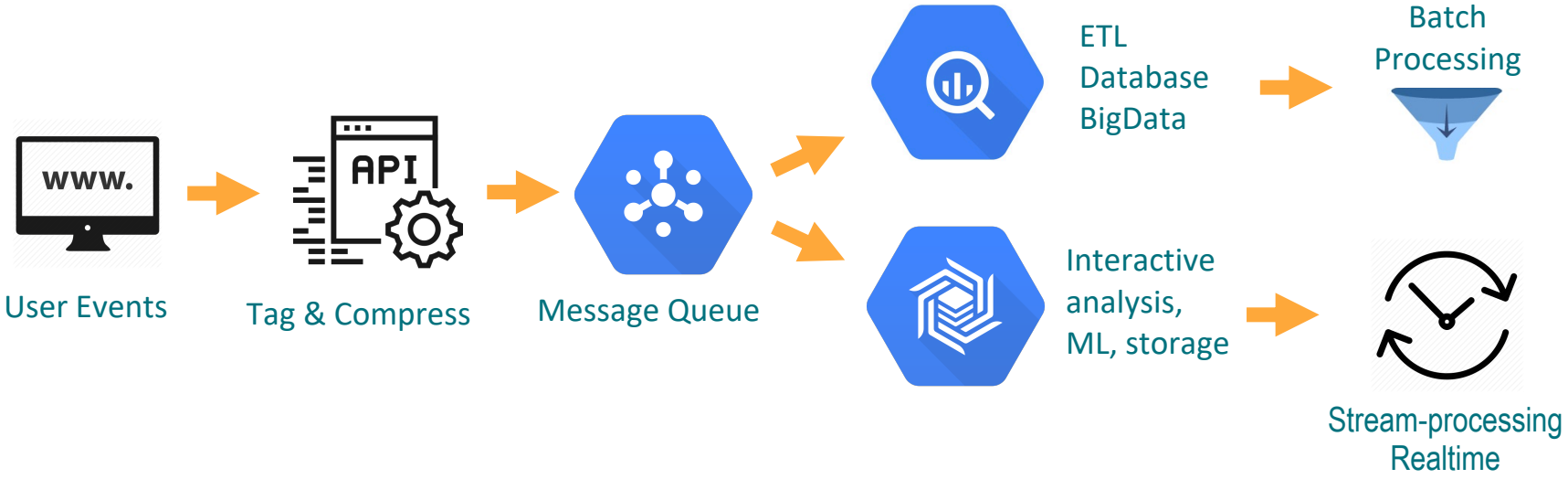
Batch Sequential *vs.* Realtime Streaming

Pass it on… Knowledge is power

Collect the data once – empower every team

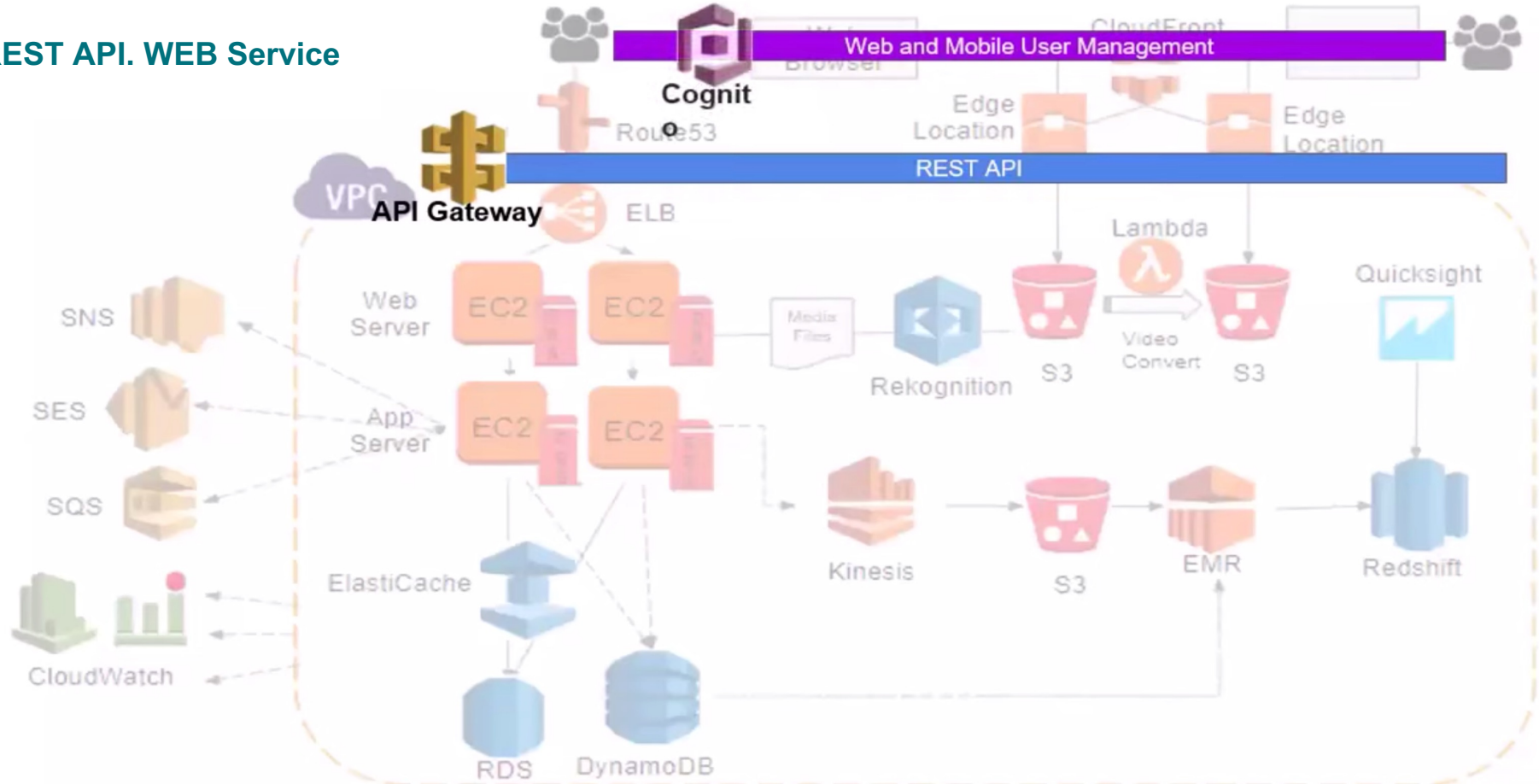
LAMBDA ARCHITECTURE

making use of both batch and stream-processing methods



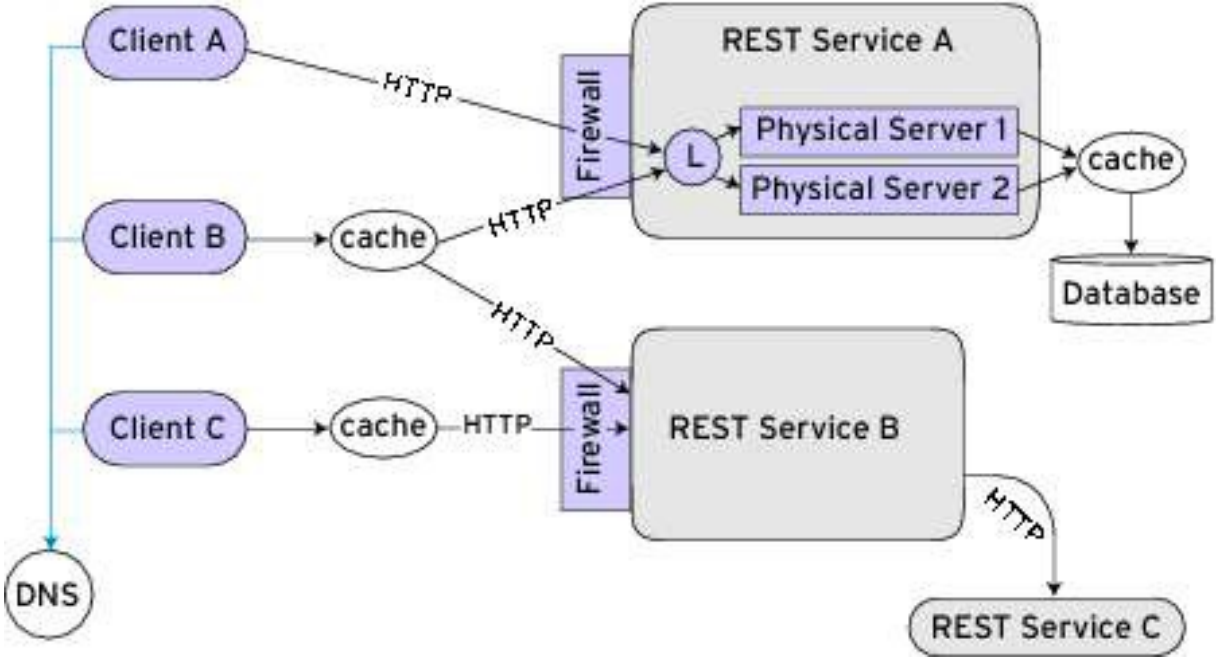
REST

REST API. WEB Service



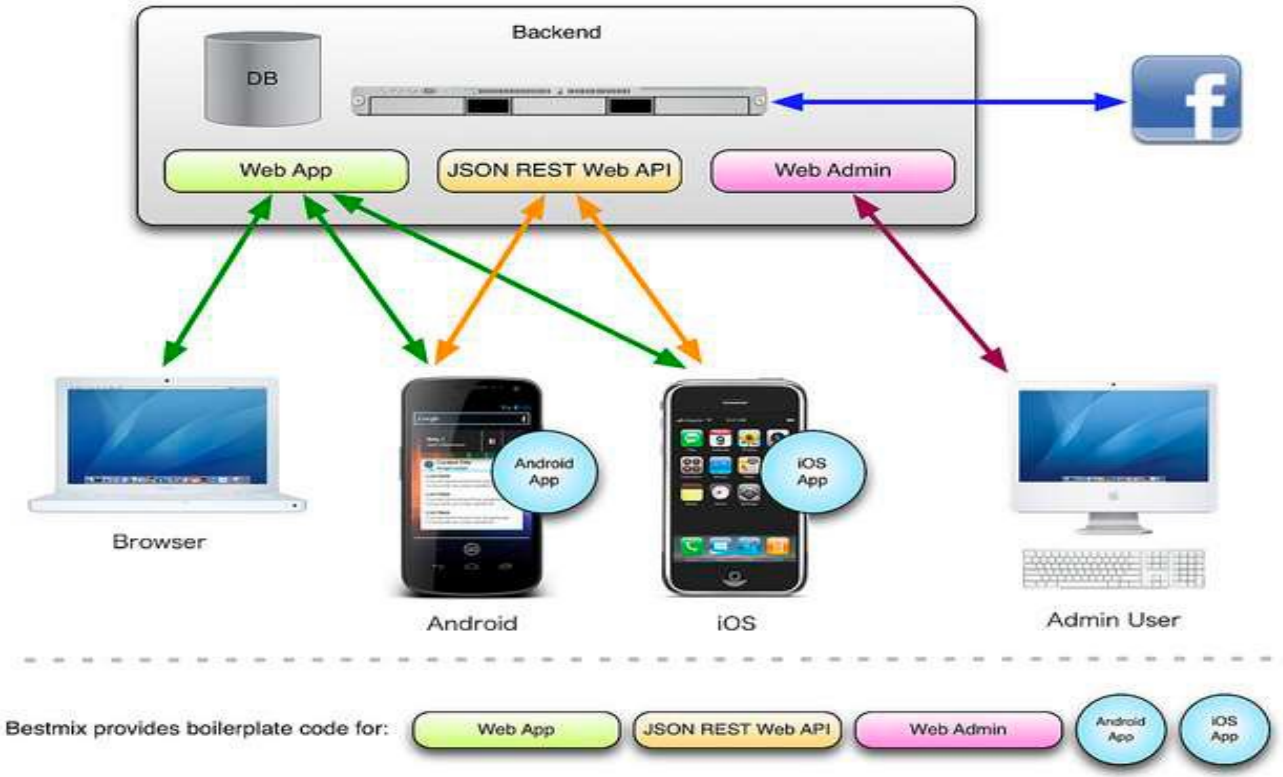
Distributed Applications

University of CINCINNATI



Distributed Web Applications

University of CINCINNATI



What is REST?

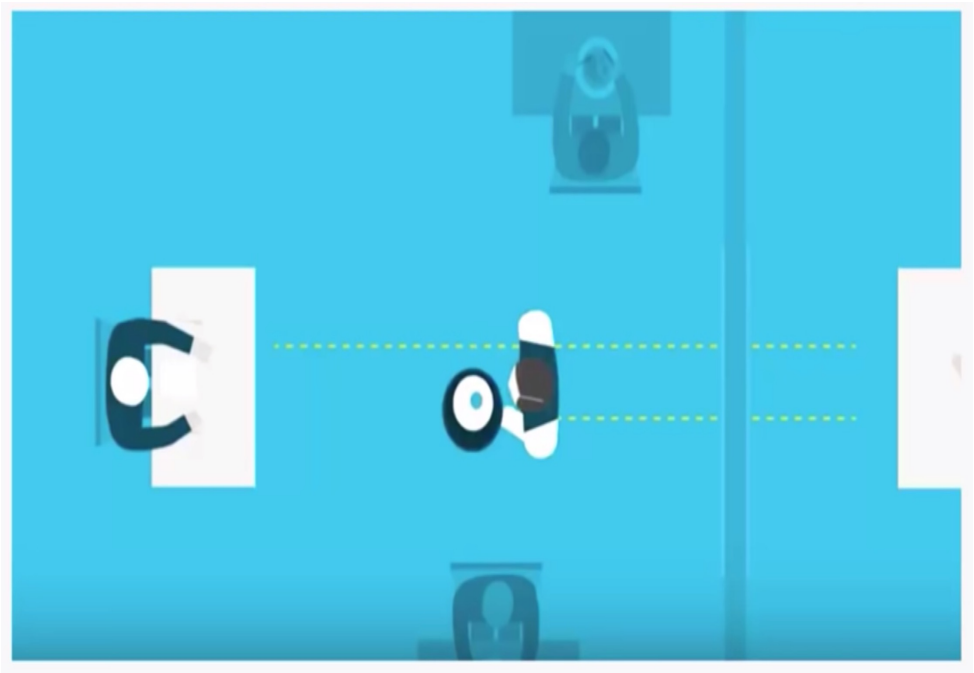
University of CINCINNATI

REST means

REpresentational

State

Transfer



Restaurant Analogy

REST

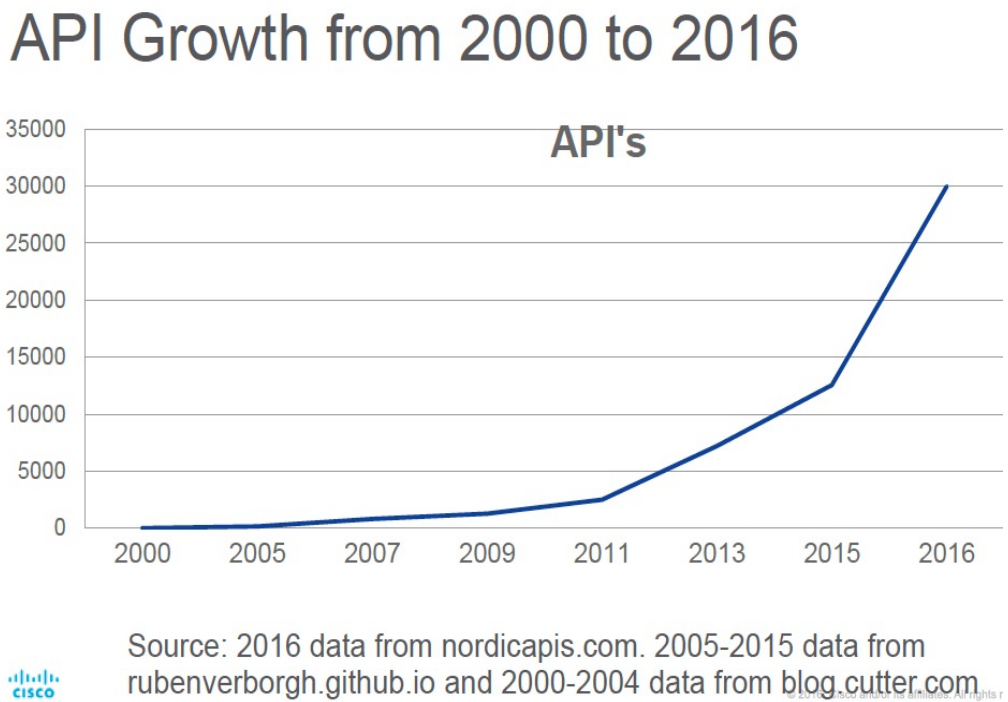
University of CINCINNATI

- **Re**presentational **S**tate **T**ransfer
- **A**rchitectural style for designing networked applications
- **REST APIs** are everywhere (**A**pplication **P**rogram **I**nterface)
- Contract provided by one piece of software to another
- Relies on a **stateless**, **client-server**, **cacheable communications protocol**
- Uses **HTTP** requests to post, read and delete data
- Treats server **objects** as **resources** that can be created or destroyed
- Can be used by **virtually** any programming language

Why REST?

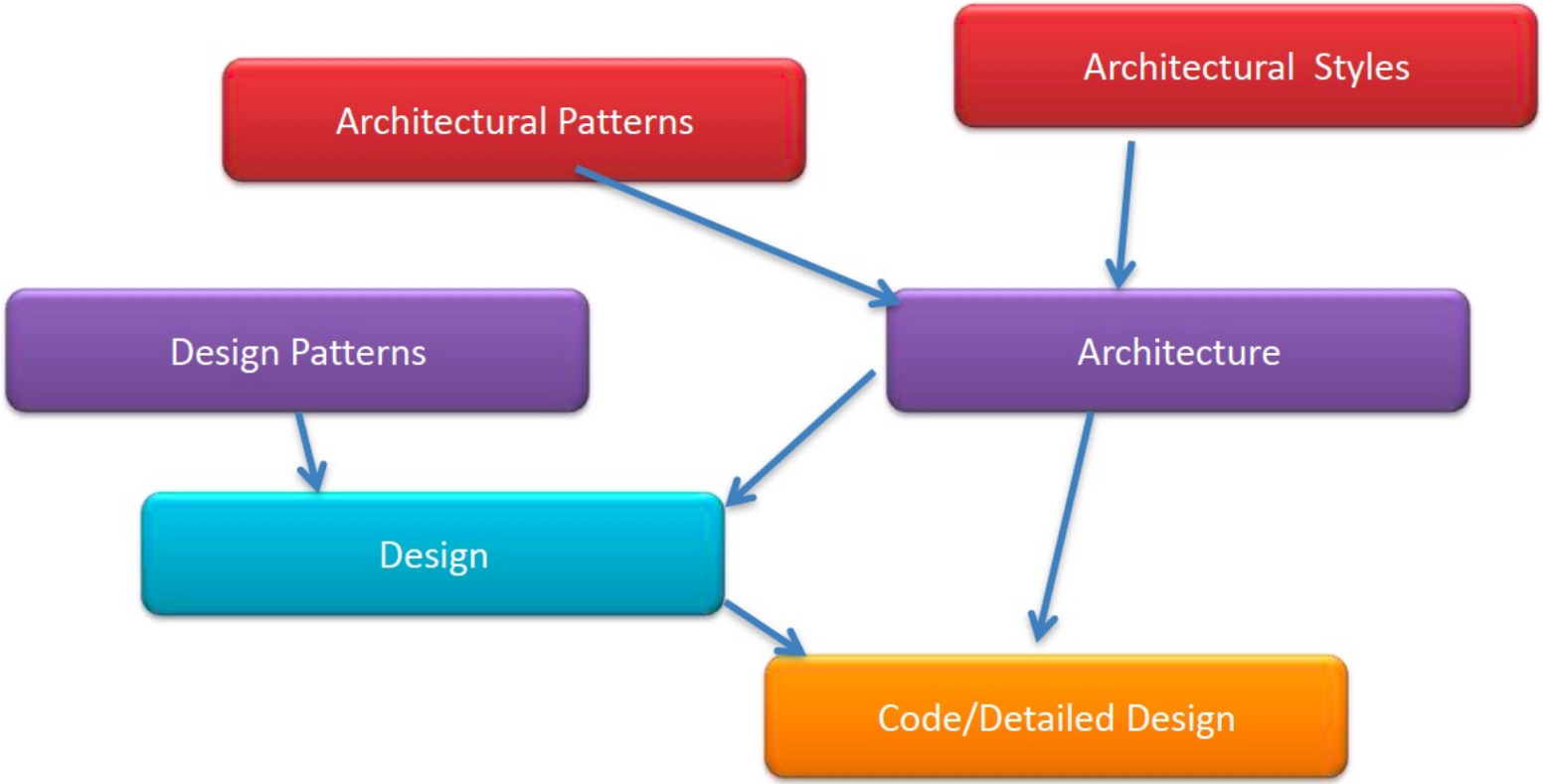
University of CINCINNATI

- Everyone/everything speaks HTTP
  - Lightweight compared to SOAP
  - Caching of resources (server decides)
  - Structured request and response
- Simplicity & ease of use
  - made for web
  - both machine & human friendly
  - scalability, modifiability, portability etc.



Making sense of the terminologies

University of CINCINNATI



Development of REST

University of CINCINNATI

"Throughout the **HTTP standardization process**, I was called on to **defend** the **design choices** of the **Web**. That is an extremely difficult thing to do within a process that accepts proposals from anyone on a topic that was rapidly becoming the center of an entire industry. I had **comments** from well over 500 developers, many of whom were distinguished engineers with decades of experience, and I had to explain everything from the most abstract notions of Web interaction **to the finest details of HTTP syntax.** That process honed my **model down to a core set of principles, properties,** and **constraints** that are now **called REST**"

-Roy Fielding

2000 PhD dissertation "Architectural Styles and the Design of Network-based Software Architectures"

What is REST?

University of CINCINNATI

REpresentational ? State ? Transfer ?

- It represent the **state of database** at a time: Static vs Dynamic
- REST is an architectural style of **networked systems** which is **based on web-standards** and the **HTTP** protocol.
- REST is best explained **as a way to talk to other machines**
- In a REST based architecture everything is a **Resource**.
- A **resource** is accessed via a **common interface based on the HTTP standard methods.**
- Typically have a REST server which provides **access** to resources, REST client which **accesses** and **modifies** the REST resources.

What is REST?

University of CINCINNATI

REpresentational ? State ? Transfer ?

- Every resource should support All the HTTP common operations.
- Resources are identified by **global IDs** (typically **URI**s or **URL**s).
- REST allows that resources have different representations, e.g., text, XML, JSON etc.
- Using **Verbs** (actions) and **Nouns** (items)
- Stateless components can be freely redeployed if something fails, and they can **scale** to accommodate load changes. **Excellent** for **distributed system**.
- This is because any request can be directed to any instance of a component.

Foundation of REST architectural style

University of CINCINNATI

- performance in component interactions, user-perceived performance, network efficiency
- scalability support of large numbers of components and interactions among components
- simplicity of a uniform interface
- modifiability of components to meet changing needs (even online)
- visibility of communication between components by service agents
- portability of components by moving program code with the data
- reliability in the resistance to failure at the system level in the presence of failures within components, connectors, or data.

REST in Action

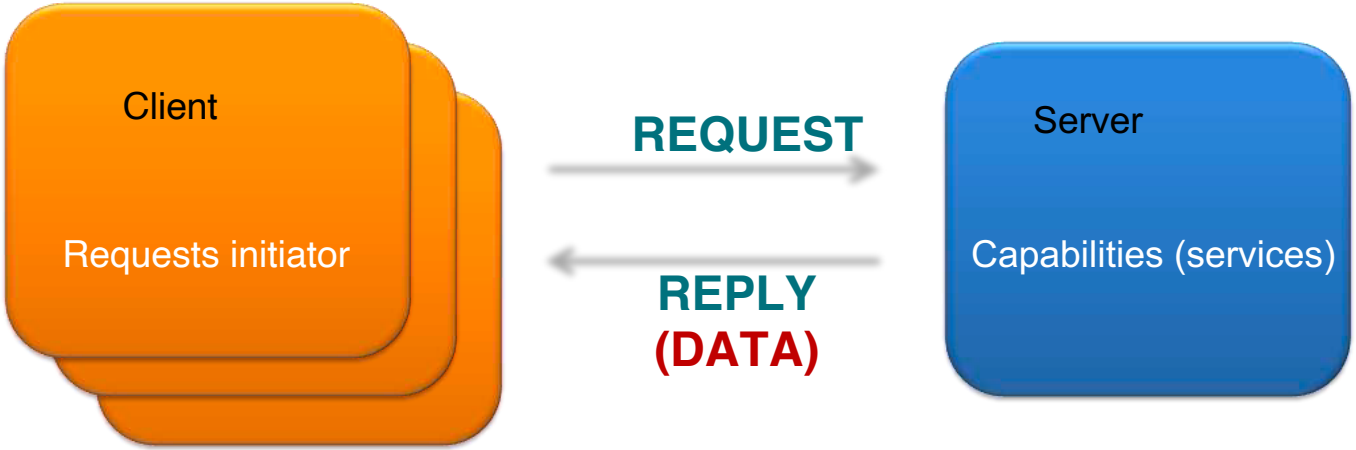
University of CINCINNATI

"It's a way for two pieces of software to talk to each other using HTTP protocol"

HTTP / HTTPS

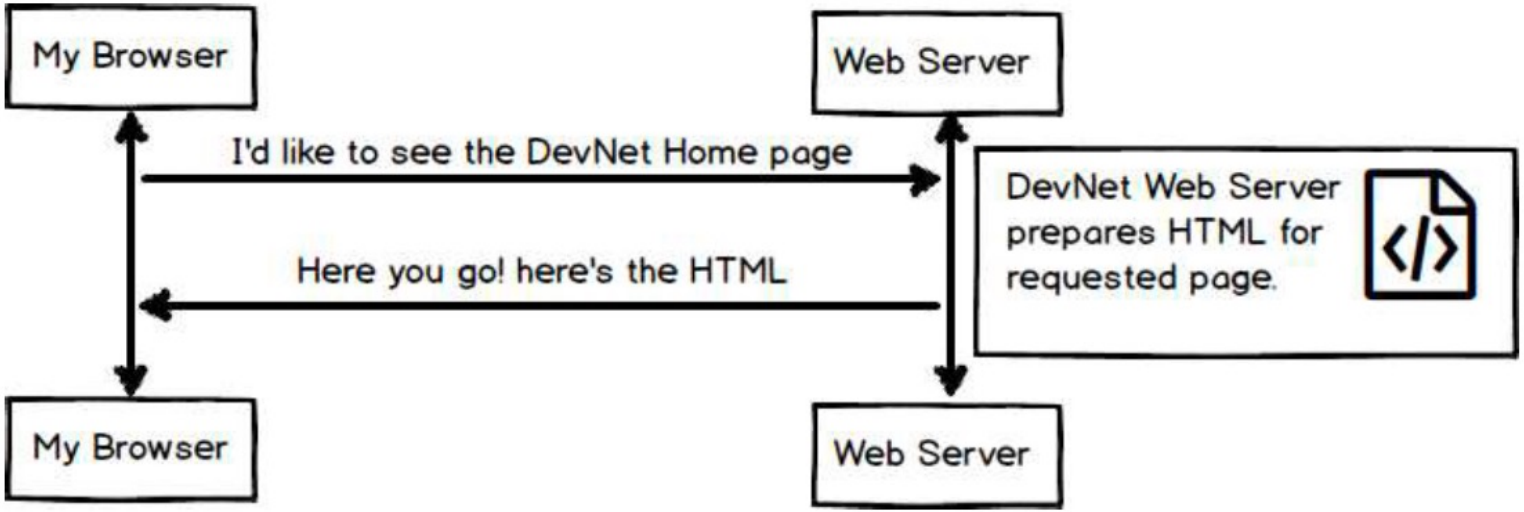
~~COBRA, RPC, SOAP~~

ACTION + RESOURCE



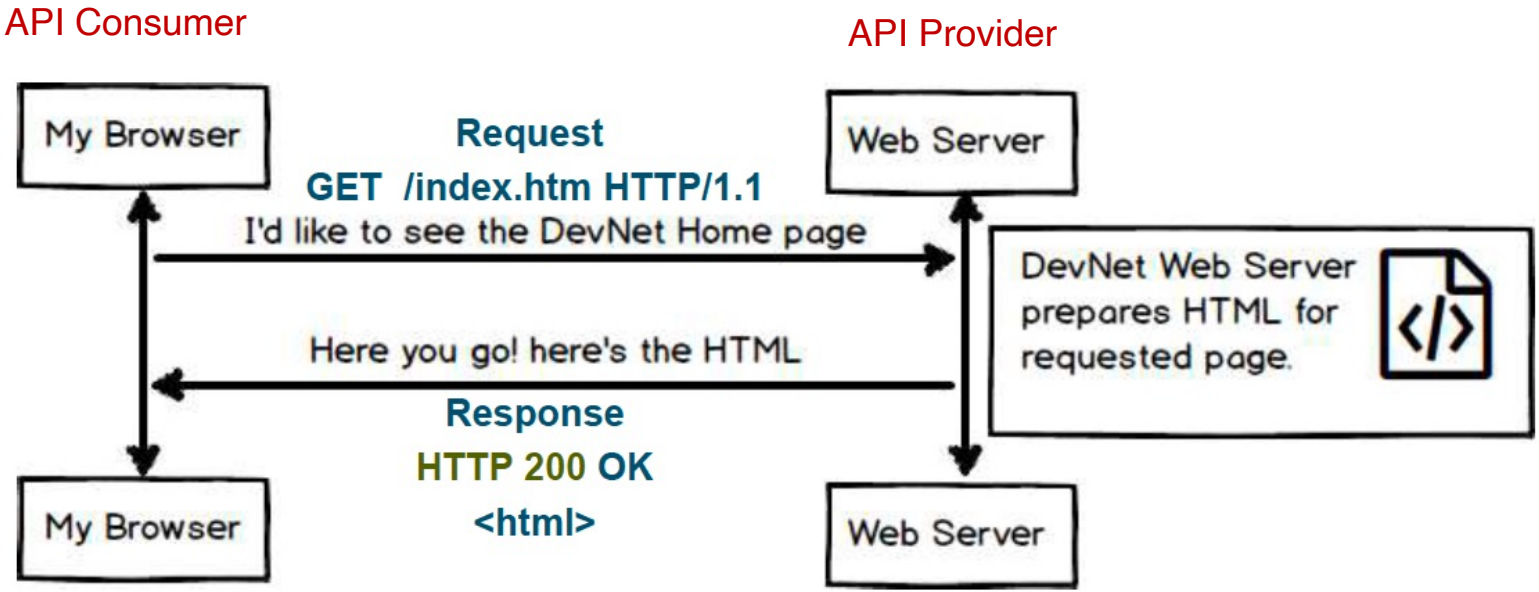
View a Web Page

University of CINCINNATI



REST APIs use Request & Response

University of CINCINNATI



REST - An Architectural Style

Elements:

- Components → Proxy, gateway, firewalls.
- Connectors → clients, servers etc.
- Data → resources, representation etc.

REST:

- Ignores component implementation details
- Focus on roles of components, their interaction and their interpretation of data elements
- Uses HTTP requests to post, read and delete data

It's a Resource-oriented Architecture

University of CINCINNATI

- **Addressability**
  - Resources are uniquely identified by URIs
  - URIs should be discoverable by clients
  - Scope of each resource should be definite

- **Statelessness**
  - No connection state maintained between REST and invocation
  - Every request executes in complete isolation on the server
  - Avoid HTTP sessions and cookies
  - Easier to scale and less side-effects

Building block for Web Apps

It's a Resource-oriented Architecture

University of CINCINNATI

- **Connectedness**
  - Resources should link together in their representations
- **Uniform Interface**
  - HTTP GET, PUT, POST, DELETE, HEAD, OPTIONS
    - Represents invocation action
  - HTTP method header
    - Returned with every request
  - HTTP status codes
    - Primary means of reporting the outcome of a REST operation

An API is like …

An API (Application Programming Interface) can be thought of as a contract between one piece of computer software to another.

HTTP Methods

The PUT, GET, POST and DELETE methods are typically used in REST based. The following table gives an explanation of these operations:

HTTP Method	CRUD Operation	Description
POST	INSERT	Addes to an existing resource
PUT	UPDATE	Overrides existing resource
GET	SELECT	Fetches a resource. The resource is never changed via a GET request
DELETE	DELETE	Deletes a resource

HTTP Status Codes

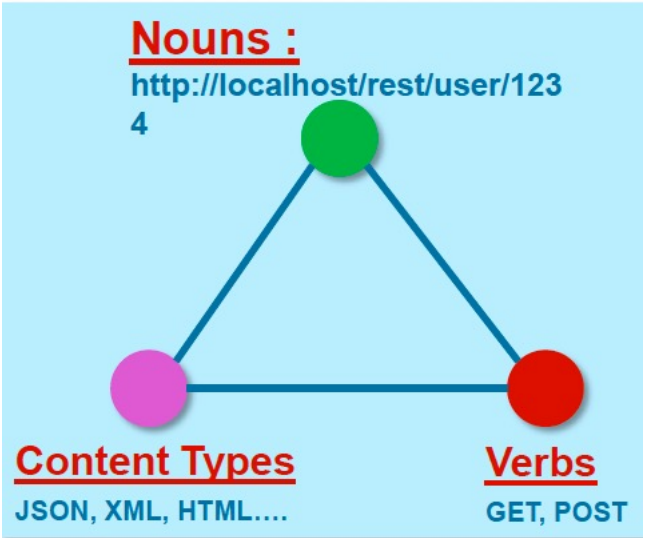
University of CINCINNATI

To Remember - Status Codes	
200 – OK	Successful Return for sync call
307 – Temporarily Moved	Redirection
400 – Bad Request	Invalid URI, header, request param
401 – Un authorized	User not authorized for operation
403 – Forbidden	User not allowed to update
404 – Not Found	URI Path not available
405 – Method not allowed	Method not valid for the path
500 – Internal Server Error	Server Errors
503 – Service unavailable	Server not accessible

REST in HTTP

University of CINCINNATI

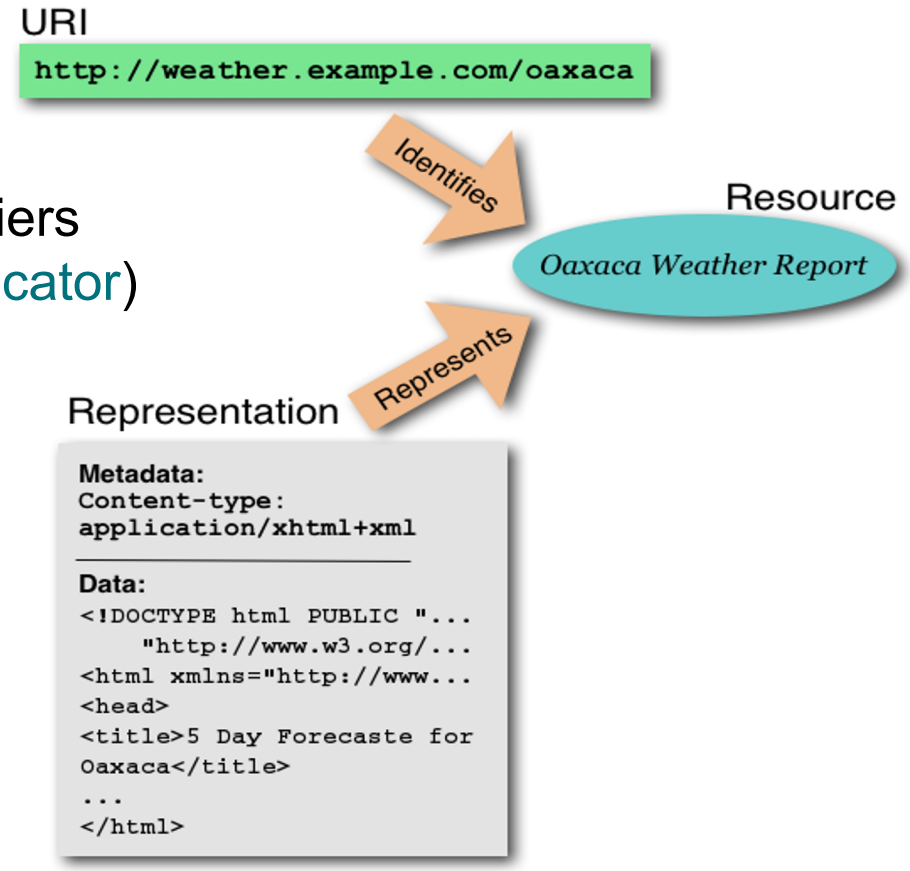
- Well defined protocol (web browsers, web servers, etc.)
  - Request and Response
  - Stateless
- Request Contents
  - URL
  - HTTP Verb
  - Header
  - Message Body



REST : Elements

University of CINCINNATI

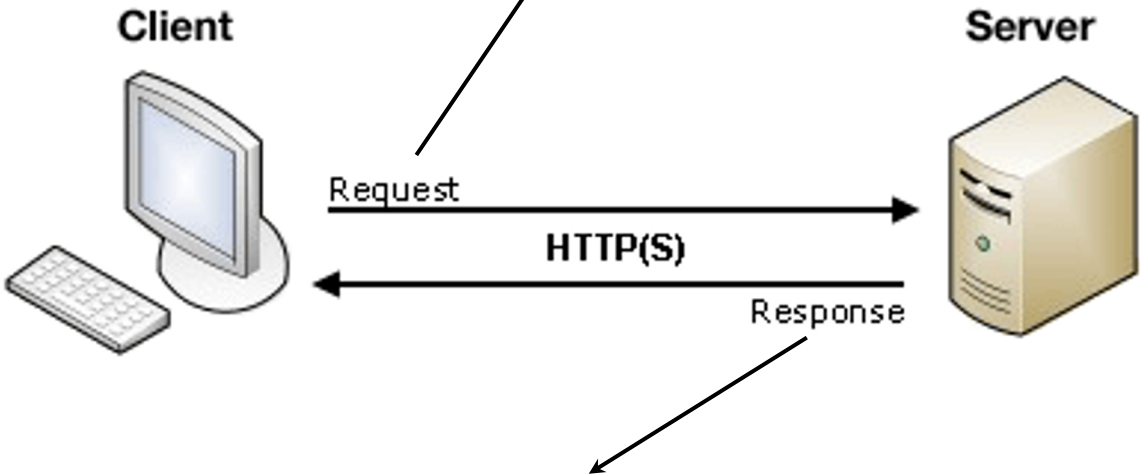
- Resource
- URI → Uniform Resource Identifiers
  (or URL → Uniform Resource Locator)
- Web Pages (HTML)



Why Representational State Transfer?

University of CINCINNATI

Request: http://weather.example.com/oaxaca



Response : OaxacaWeather.html

Definition

University of CINCINNATI

"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 an 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."

-Roy Fielding

An Architectural Style of Networked System

- Underlying Architectural model of the world wide web.
- Guiding framework for Web protocol standards.

REST based web services:

- Online shopping
- Search services
- Dictionary services

Example : Online Book Store

University of CINCINNATI

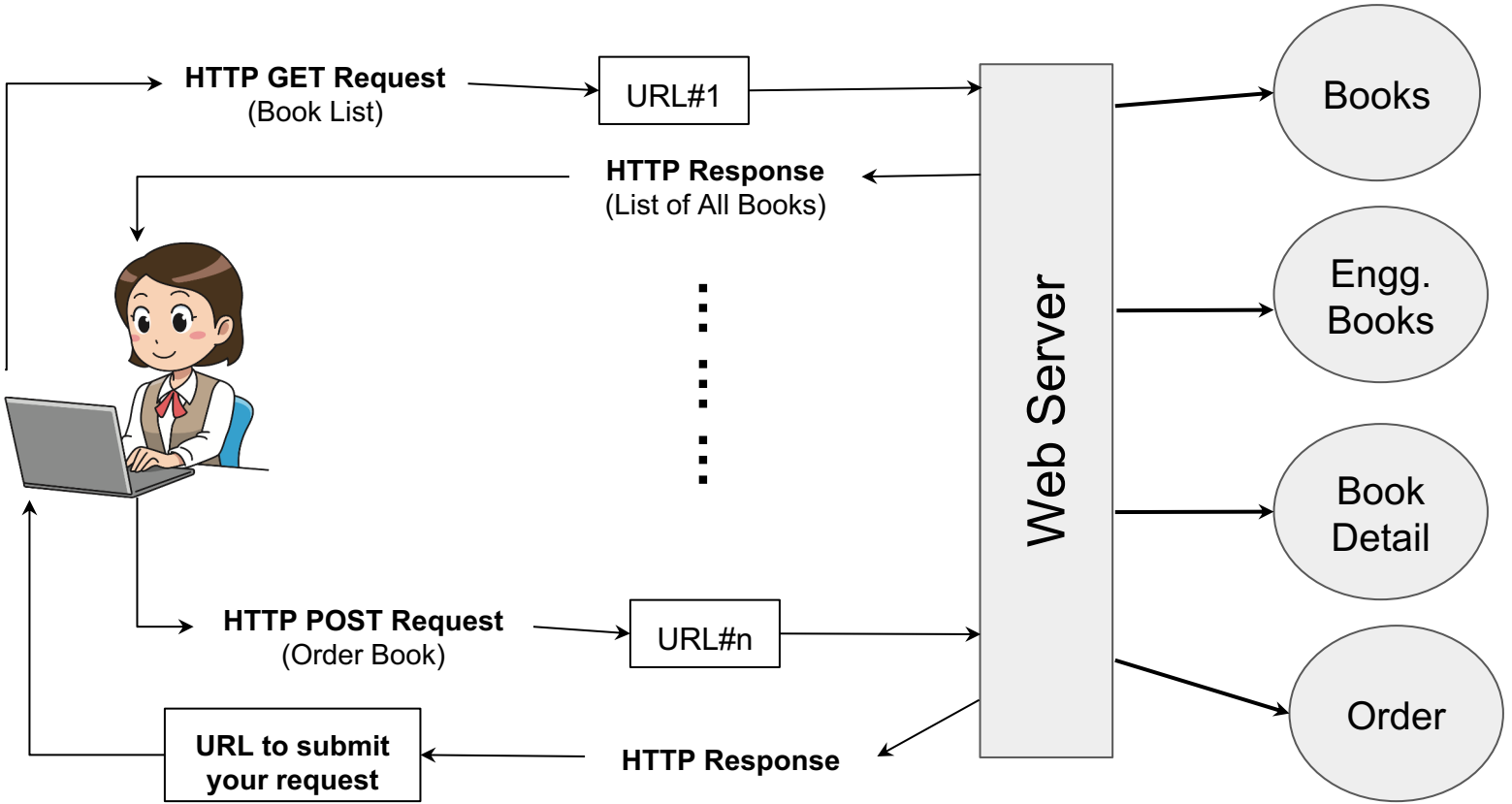
Book Store Web Services

Bookstore Inc has deployed some web services to enable students to:

- ➔ Get a list all books
- ➔ Get list of all books related to a specific field of study
- ➔ Get details of a book
- ➔ Submit purchase order for that book

REST : Implementing the Web Services

University of CINCINNATI



Service → Get Book List

University of CINCINNATI

Web service makes available a URL to books

Client Uses: https://www.books.com/dept

Documents received by Client:

```
<?xml version="1.0"?>
<p:Dept xmlns:p="http://www.books.com" xmlns:xlink="http://www.w3.org/1999/xlink">
        <Dept id="Engg" xlink:href="http://www.books.com/dept/Engg"/>
        <Dept id="Engl" xlink:href="http://www.books.com/dept/Engl"/>
        <Dept id="Hist" xlink:href="http://www.books.com/dept/Hist"/>
        <Dept id="Finc" xlink:href="http://www.books.com/dept/Finc"/>
</p:Dept>
```

Service → Get Book Details

University of CINCINNATI

Web service makes available a URL to book details

Client Uses: https://www.books.com/dept/Engg/04557

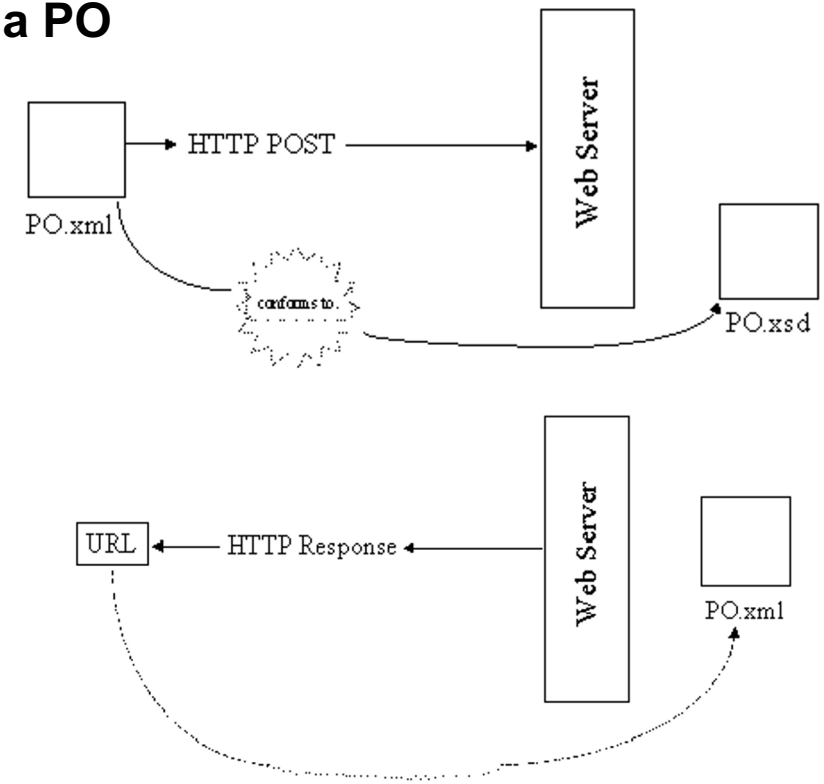
Documents received by Client:

```
<?xml version="1.0"?>
<p:Dept xmlns:p="http://www.books.com/dept" xmlns:xlink="http://www.w3.org/1999/xlink">
        <p:Engg xmlns:p="http://www.books.com/dept/Engg"  xmlns:xlink="… … … ">
                        <Book-ID>04557</Book-ID>
                        <Name>Cloud Computing</Name>
                        <Description>This is Cloud Computing Text Book</Description>
                        <Specification
xlink:href="http://www.books.com/dept/Engg/04557/specification"/>
                <UnitCost currency="USD">99.10</UnitCost>
                        <Quantity>10</Quantity>
                </p:Stream>
</p:Dept>
```

Service → Submit Purchase Order

University of CINCINNATI

Web service makes available a URL to submit a PO



1. Client creates a PO instance document (PO.xml
2. Submit the PO.xml (HTTP POST)) using **XML Schema** Definition format (PO.xsd**)**
3. PO service responds with a URL to the submitted PO.

Interpretation – Resource Manager

http://www.books.com/api/customers

Interpretation - Resource Manager

http:// www.books.com/api/customers

Secure channel HTTPS

Interpretation – Resource Manager

http://www.books.com/api/customers

Domain Name

Interpretation – Resource Manager

http://www.books.com/api/customers

Not compulsory but customary convention to expose restful services include the word API somewhere in the Address. It can be after the domain or can be a subdomain like API.movies4all

- Session details
  - URI (ID)
  - State (start/end/status etc.)

Interpretation – Resource Manager

http://www.books.com/api/customers

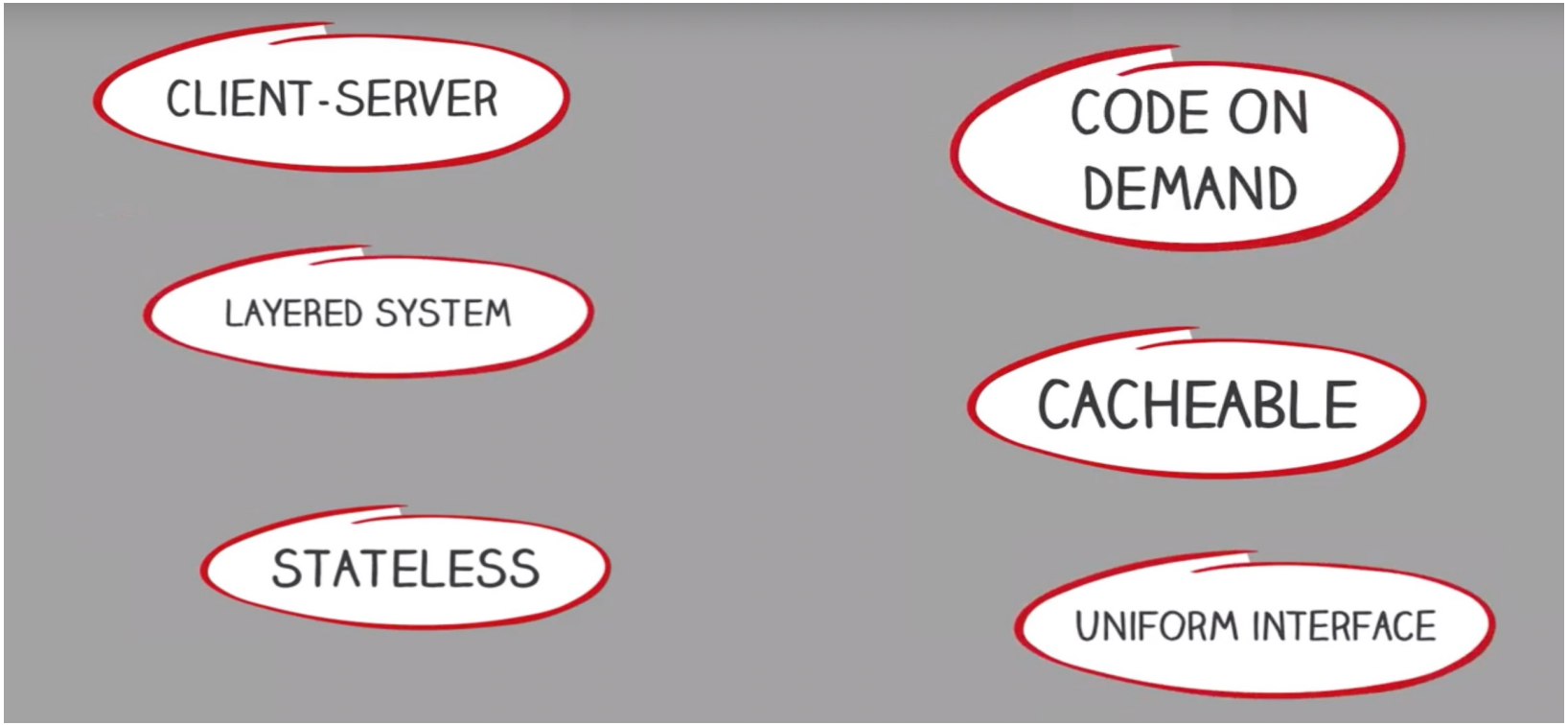
Resource, or collection of in the application. Expose resources such as customers movies rentals where operations such as select, create, updates to the customers can be done by sending an HTTP requests.

- Resources
  - Knows status
  - URI for the Resource representation on the RM
  - URI for the Resource itself

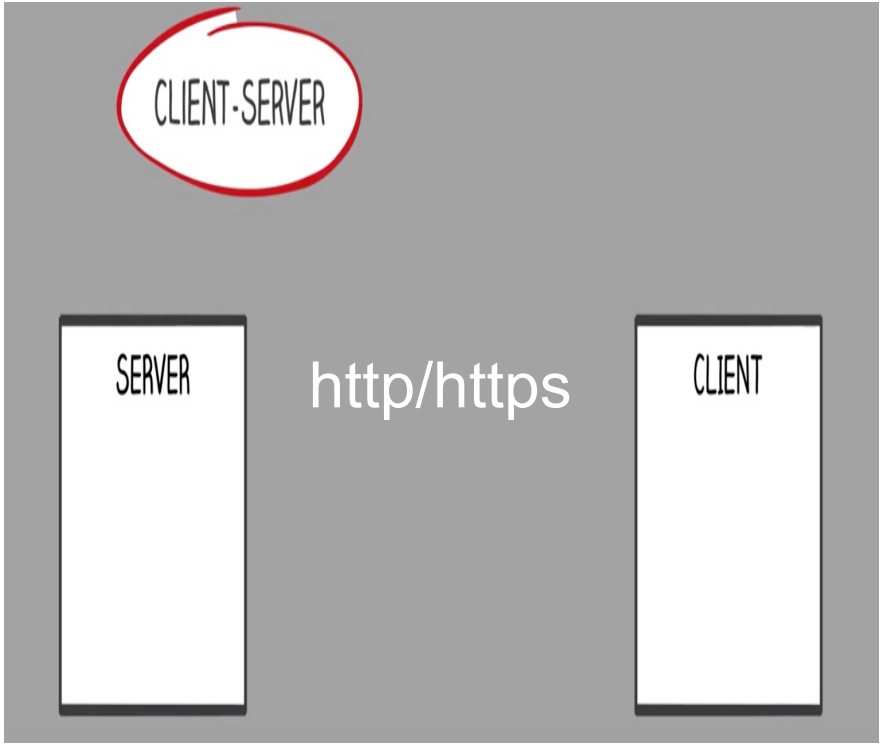
RESTful Architectural principles

University of CINCINNATI

Six guiding constraints define a RESTful system

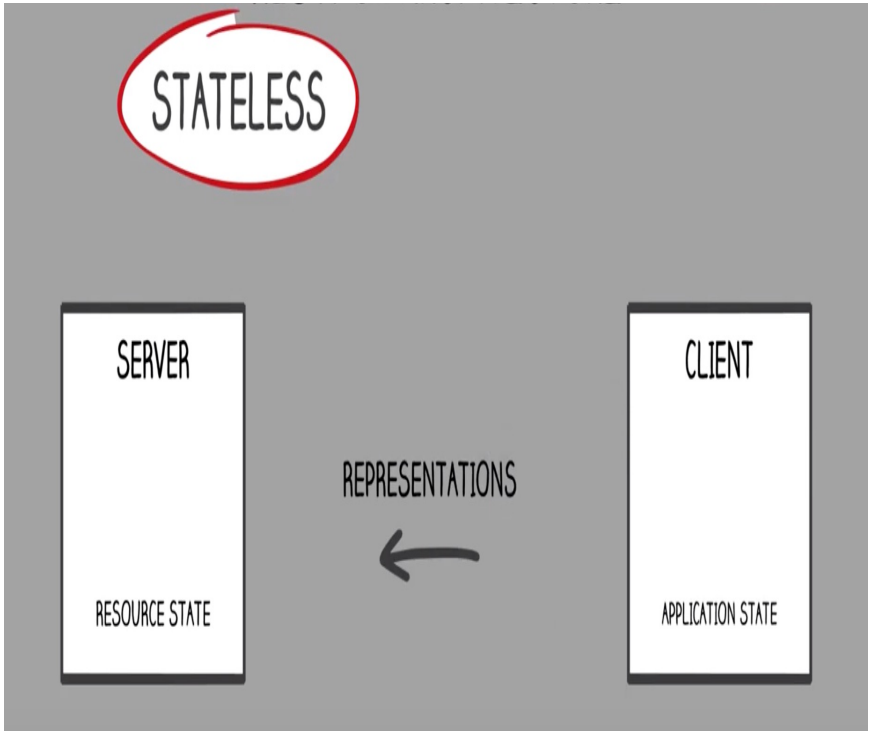


Client-server architecture



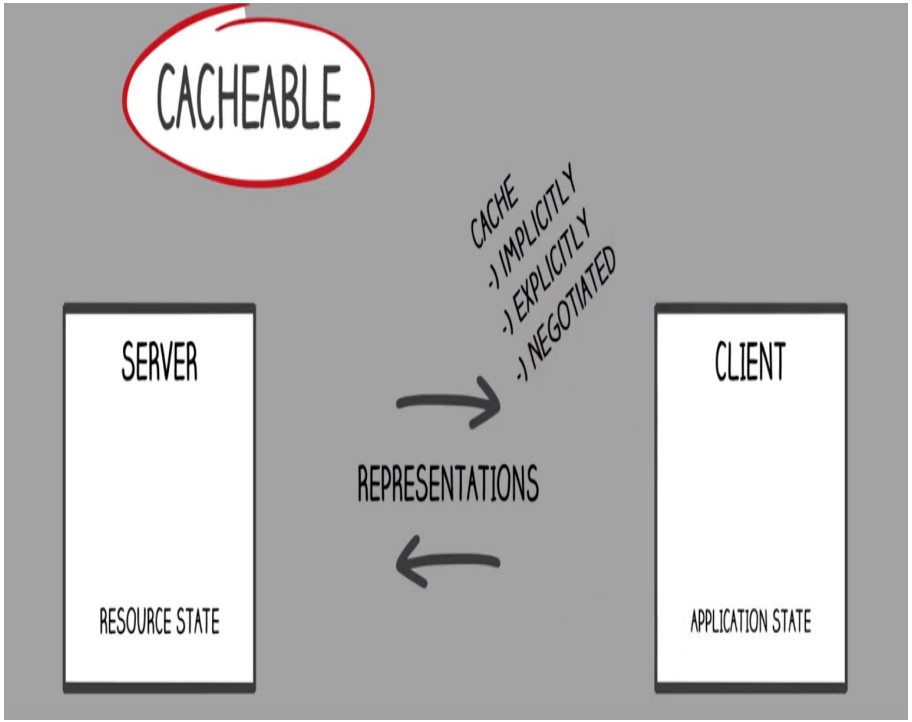
- separation of concerns
- distributed and disconnected components
- encourages components to evolve independently
- provides much more:
  - scalability
  - maintainability
  - portability

Stateless protocol



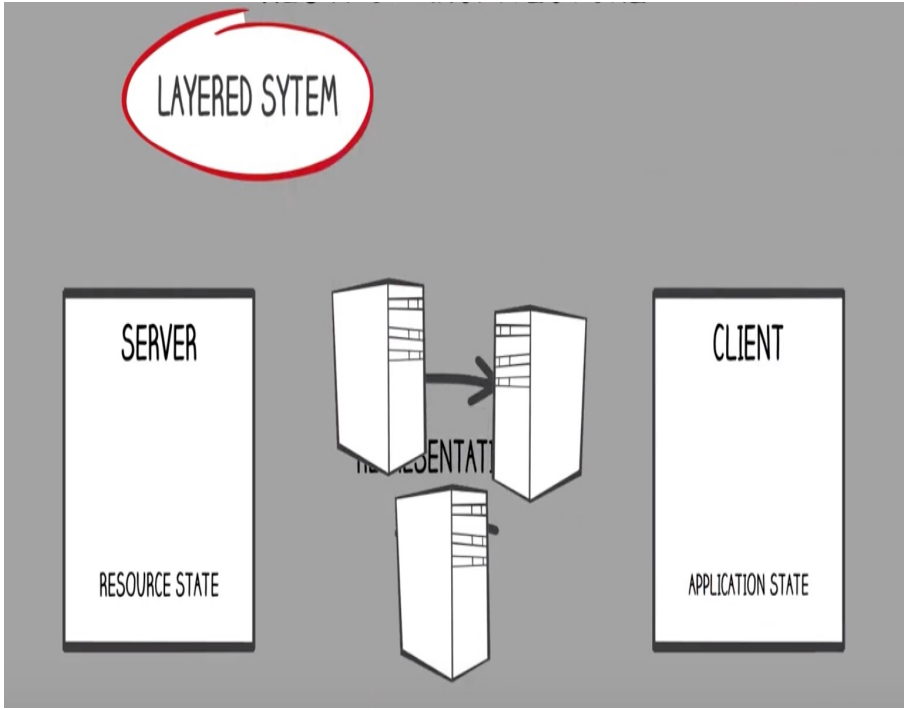
- no client's state is stored on the server
- every request is self contained & self descriptive i.e all requests from clients contain all the information necessary for the server to provide service.
- session state is kept entirely on the client

Web cache



- responses should be cacheable
  - implicitly
  - explicitly
  - negotiated

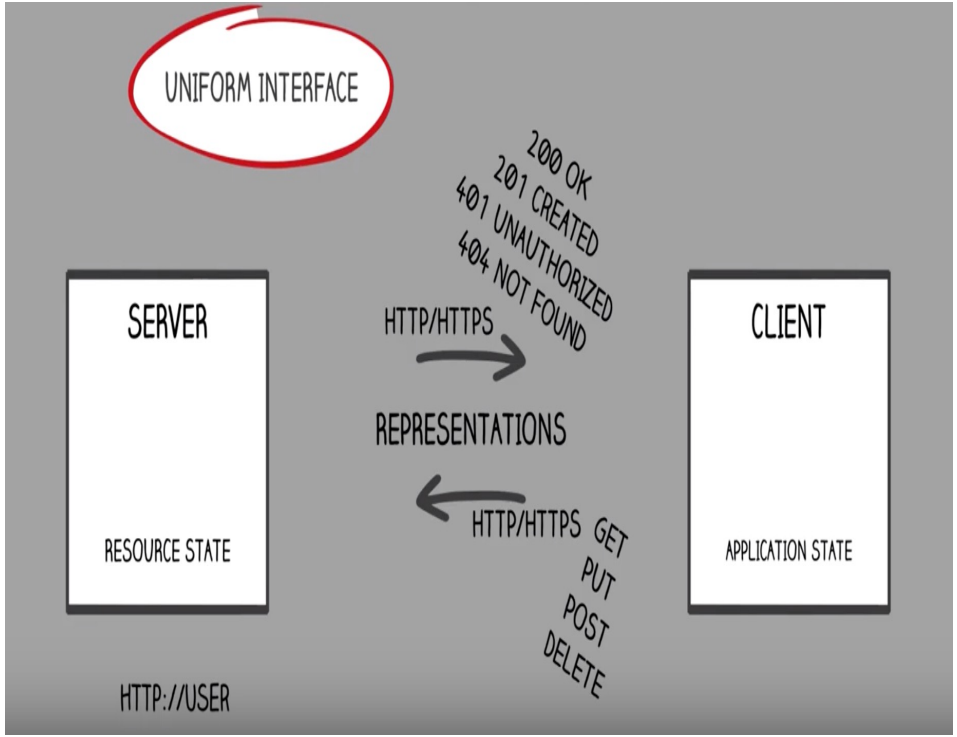
Layered system



- there could be different layers in between the client and server like proxies, middleware, services, caches, load balancers etc.
- client doesn't know how the response is being generated, and is not concerned either (separation of concern again)

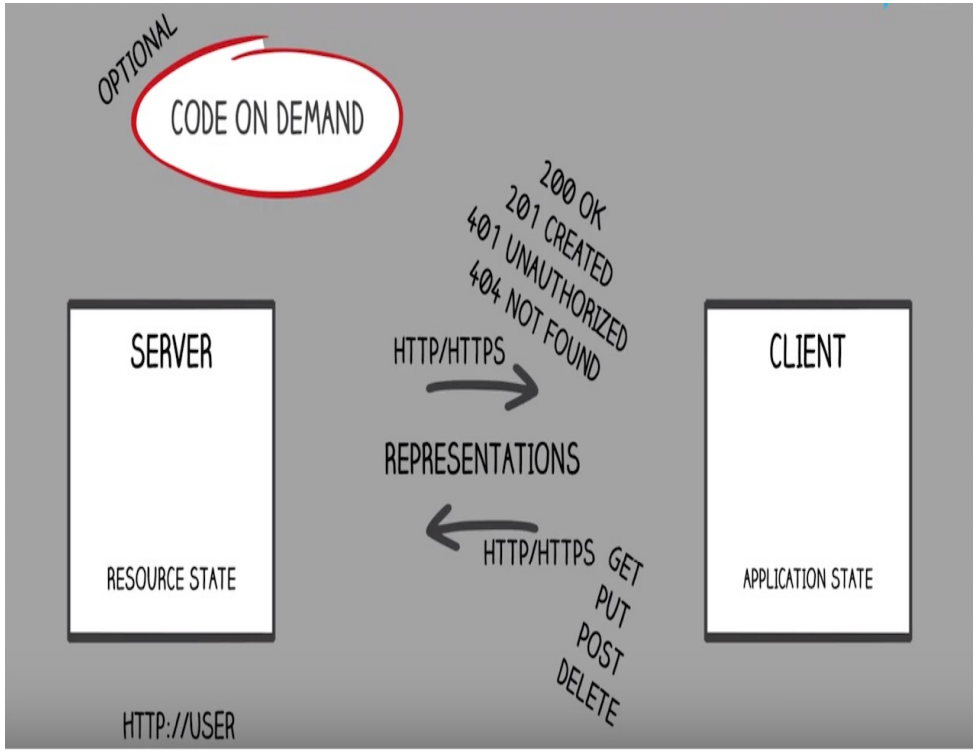
Uniform interface

University of CINCINNATI



- uniform interface between the client and server
- simplification & decoupling of the architecture
- in HTTP,
  - URIs - represent resources
  - http verbs (GET / POST / PUT / DELETE) - represent actions
  - http response, status code, headers

Client-side scripting



OPTIONAL

- Servers can temporarily extend or customize the functionality of a client by transferring executable code: for example, compiled components such as Java applets, or client-side scripts such as JavaScript.

Web Services

University of CINCINNATI

Let's Look At An Example

Get customer

Request

```
GET https://www.icici.com/api/customer
```

Response

```
[
    {id: 1, name: '' },
    {id: 2, name: '' },
    ...
]
```

Get A customer

University of CINCINNATI

Request

```
GET https://www.icici.com/api/customer/1
```

Response

```
{id: 1, name: '' },
```

Update a customer

University of CINCINNATI

Request

```
PUT https://www.icici.com/api/customer/1

{ name: '' }
```

Response

```
{id: 1, name: '' },
```

Create a customer

Request

```
POST https://www.icici.com/api/customer/2

{ name: '' }
```

Response

```
{id: 2, name: '' },
```

Resources in REST

University of CINCINNATI

Resources

As pretty much every representation of a resource will contain some links (at least the `self` one) we provide a base class to actually inherit from when designing representation classes.

```
class PersonResource extends ResourceSupport {

  String firstname;
  String lastname;
}
```

Inheriting from `ResourceSupport` will allow adding links easily:

```
PersonResource resource = new PersonResource();
resource.firstname = "Dave";
resource.lastname = "Matthews";
resource.add(new Link("http://myhost/people"));
```

This would render as follows in JSON:

```
{ firstname : "Dave",
  lastname : "Matthews",
  links : [ { rel : "self", href : "http://myhost/people" } ] }
```

REST API

The URI/URL where api/service can be accessed by a client application

GET	https://mysite.com/api/users	
GET	https://mysite.com/api/users/1	OR https://mysite.com/api/users/details/1
POST	https://mysite.com/api/users	
PUT	https://mysite.com/api/users/1	OR https://mysite.com/api/users/update/1
DELETE	https://mysite.com/api/users/1	OR https://mysite.com/api/users/delete/1

CURL

Some API's require authentication to use their service. This could be free or paid

```
curl -H "Authorization: token OAUTH-TOKEN" https://api.github.com
```

```
curl https://api.github.com/?access_token=OAUTH-TOKEN
```

```
curl 'https://api.github.com/users/whatever?client_id=xxxx&client_secret=yyyy'
```

Example using github

University of CINCINNATI

developer.github.com/v3/

All API access is over HTTPS, and accessed from `https://api.github.com`. All data is sent and received as JSON.

```
curl -i https://api.github.com/users/octocat/orgs
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 12 Oct 2012 23:33:14 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Status: 200 OK
ETag: "a00049ba79152d03380c34652f2cb612"
X-GitHub-Media-Type: github.v3
X-RateLimit-Limit: 5000
X-RateLimit-Remaining: 4987
X-RateLimit-Reset: 1350085394
Content-Length: 5
Cache-Control: max-age=0, private, must-revalidate
X-Content-Type-Options: nosniff
```

GitHub

https://developer.github.com/v3/

https://developer.github.com/v3/users/

Register a new OAuth application

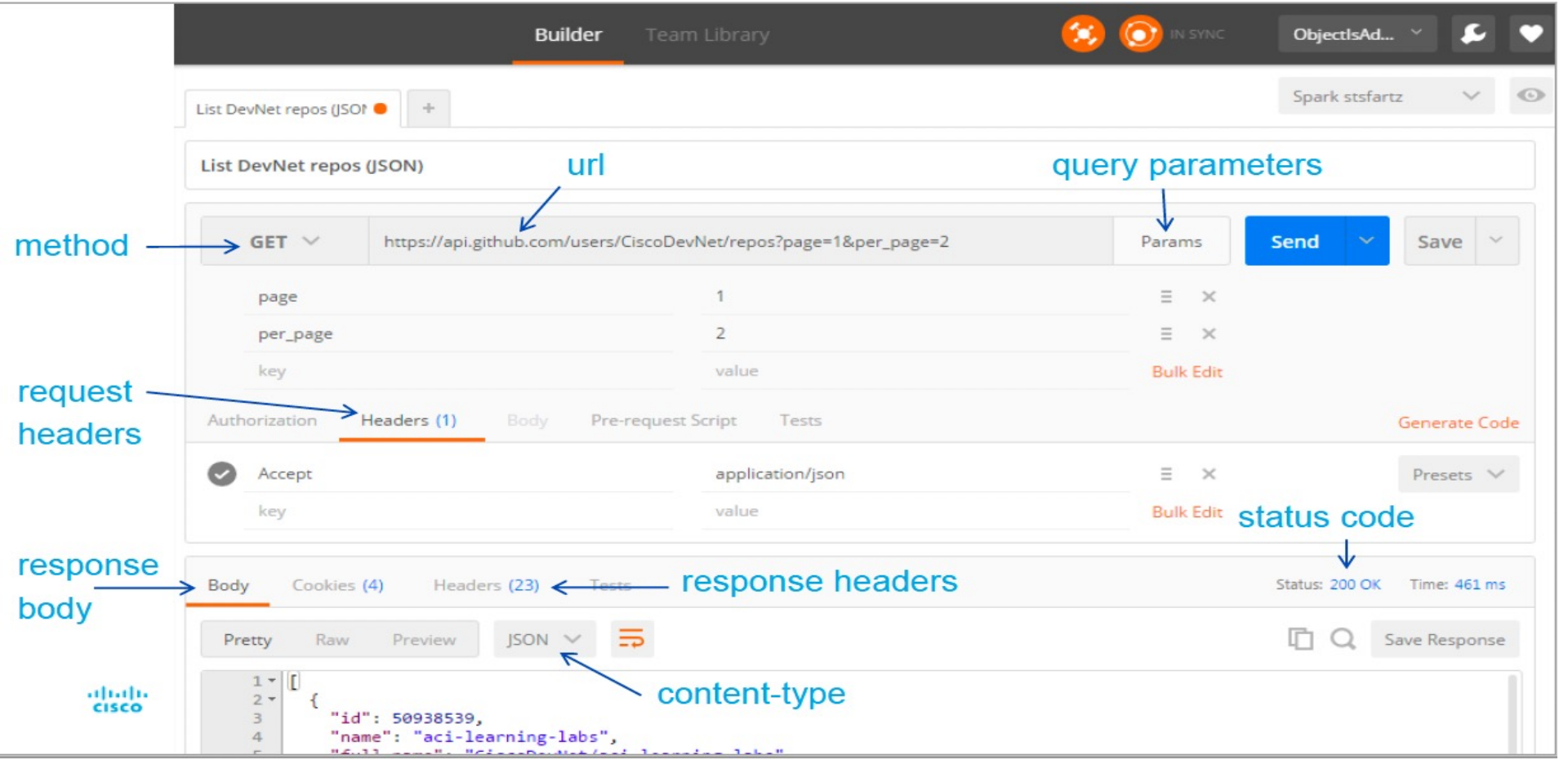
https://github.com/settings/applications/new

POSTMAN!!!!!

University of CINCINNATI

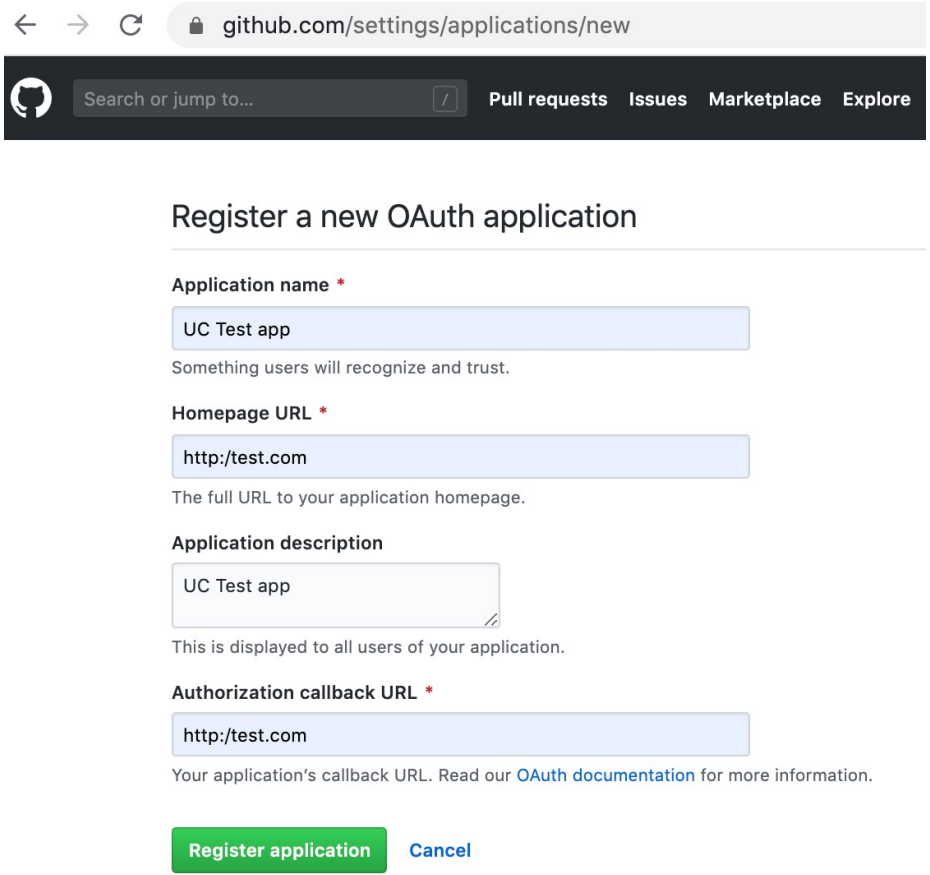
POSTMAN!!!!!

University of CINCINNATI



POSTMAN!!!!!

University of CINCINNATI



https://github.com/settings/applications/1214161

Postman: GET https://api.github.com/users/rashmikasa?client_id=596c15845748f7c48b5b&client_secret=4f8613cb0518f1c8ba88f7bc5e057d6677124925

Cmd>

curl -i 'https://api.github.com/users/rashmikasa?client_id=596c15845748f7c48b5b&client_secret=4f8613cb0518f1c8ba88f7bc5e057d6677124925'

Example using github

University of CINCINNATI

GitHub
https://developer.github.com/v3/
https://developer.github.com/users/
https://developer.github.com/users/**rashmikasa**
https://api.github.com/users/rashmikasa?client_id=596c15845748f7c48b5b&client_secret=4f8613cb0518f1c8ba88f7bc5e057d6677124925

CURL: <Command Line | Terminal>
curl -i 'https://api.github.com/users'
curl -i 'https://api.github.com/users/rashmikasa'
curl -i 'https://api.github.com/users/rashmikasa?client_id=596c15845748f7c48b5b&client_secret=4f8613cb0518f1c8ba88f7bc5e057d6677124925'

Postman: GET REST Method
https://api.github.com/users
https://api.github.com/users/rashmikasa
https://api.github.com/users/rashmikasa?client_id=596c15845748f7c48b5b&client_secret=4f8613cb0518f1c8ba88f7bc5e057d6677124925

Register a new OAuth application
https://github.com/settings/applications/**new**
UC Test app
http://test.com
UC App
http://test.com

https://www.diigo.com/user/jfjcn1/webApi
https://blog.diigo.com/category/tips-of-the-day/

Summary

University of CINCINNATI

- Architectural Style based on HTTP
- Uniform interface: all resources are accessed with a generic
- 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.
- Resources (Things)
- Verbs (Actions)
    - GET, POST, PUT, DELETE, etc.
- Response gives a representation of the resource (XML, JSON, image, etc.)

More Info: http://en.wikipedia.org/wiki/REST

Constraints to be mindful

University of CINCINNATI

- 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.

Principles & Best Practices

University of CINCINNATI

- Identify all the conceptual entities that we wish to expose as services. (Examples we saw include resources such as : dept list, detailed book data, purchase order)
- Create a URL to each resource.
- 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).
- 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 & Best Practices

University of CINCINNATI

- ➔ Put hyperlinks within resource representations to enable clients to drill down for more information, and/or to obtain related information.
- ➔ Design to reveal data gradually. Don't reveal everything in a single response document. Provide hyperlinks to obtain more details.
- ➔ 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.
- ➔ Describe how our services are to be invoked using either a WSDL document, or simply a HTML document.

REST METHODS

University of CINCINNATI

Methods	CRUD	Example
GET	Fetch all or any resource	GET /user/ - Fetch all GET /user/1 – Fetch User 1
POST	Create a Resource	POST /user?name=user1&age=20
PUT	Update a Resource	PUT /user/1? name=changed-user1&age=22
DELETE	Delete a Resource	DELETE /user/1
HEAD	Fetch Metainfo as header	HEAD /user
OPTIONS	Fetch all VERBS allowed	OPTIONS /user

Finally What is REST?

University of CINCINNATI

Still confused about REST?

University of CINCINNATI

It is just …
… some Theory
… give it a REST