



# INFORMATION TECHNOLOGIES

TECH  
DAYS

REST  
**Refresher**

Now  
with  
**REST!**



Greg Gómez  
Web Developer  
UNM IT  
gogogo@unm.edu

# Overview Of Today's Presentation

- HTTP
- REST







# HTTP

- HyperText Transfer Protocol

# HTTP



“HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access.... HTTP was developed to facilitate hypertext and the World Wide Web.”

[https://en.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)

# HTTP

- But it's used for more than just delivering webpages
- APIs
- DevOps
  - Eg: git
- **REST**

# HTTP - History

- Tim Berners-Lee invented HTTP in 1989.
- First documented in 1991 (V0.9).
- V1.0 - 1996
- V1.1 - 1997
- V2.0 - 2015



# HTTP - Background

- While getting a page, a web browser will typically make many HTTP requests.
- We'll be focusing on individual HTTP transactions, not on web pages.

# Caveat

- I'll be doing some hand waving through network connections, encrypted connections (SSL), binary files, streaming, etc.

# HTTP

- Please note that in the following slides, the transactions are in plain text!

# HTTP - Request - Format

```
METHOD path HTTP Version ↵
```

```
Headers... ↵
```

```
↵
```

```
Body... ↵
```

```
↵
```



# HTTP – Request - Example

```
GET / HTTP/1.1 ↵
```

```
Host: www.php.net ↵
```

```
↵
```

# HTTP – Response - Format

```
<HTTP Version> <Status Code> <Status Message>\r\n
```

```
Headers... \r\n
```

```
\r\n
```

```
Body... \r\n
```

```
\r\n
```

# HTTP – Response - Example

```
HTTP/1.1 200 OK↵
```

```
Date: Mon, 08 Oct 2018 20:49:08 GMT↵
```

```
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips↵
```

```
X-Powered-By: PHP/7.1.10↵
```

```
↵
```

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
```

```
<head>
```

```
...
```



Tardigrades, also known colloquially as water bears, or moss piglets are a phylum of water-dwelling, eight-legged, segmented micro-animals.



# HTTP - Demonstration

- Connect to a web site using telnet
- Issue a proper Request
- Examine the Response
- Let's go!

# HTTP - Demonstration

- Oops, first, what's telnet?
- A protocol that allows a user on one computer to connect to another computer and send **plain text** between them.
- Client/server architecture.
- It was *the* way to connect to remote machines for a long time.
- But it's insecure, and has largely been replaced by ssh

# HTTP - Demonstration

- That's all there is to it.
- Again: plain text.
- Simple.
- At least though HTTP 1.1...

# HTTP – Request Methods

- GET
- HEAD
- POST
- PUT
- DELETE
- CONNECT
- OPTIONS
- TRACE



# HTTP – Request Methods - GET

The GET method requests transfer of a current selected representation for the target resource.

# HTTP – Request Methods - HEAD

The HEAD method is identical to GET except that the server **MUST NOT** send a message body in the response (i.e., the response terminates at the end of the header section).

<https://tools.ietf.org/html/rfc7231#section-4>

# HTTP – Request Methods - POST

The POST method requests that the target resource process the representation enclosed in the request according to the resource's own specific semantics.

<https://tools.ietf.org/html/rfc7231#section-4>

# HTTP – Request Methods - PUT

The PUT method requests that the state of the target resource be created or replaced with the state defined by the representation enclosed in the request message payload.

<https://tools.ietf.org/html/rfc7231#section-4>

# HTTP – Request Methods - DELETE

The DELETE method requests that the origin server remove the association between the target resource and its current functionality.

<https://tools.ietf.org/html/rfc7231#section-4>



# HTTP – Request Methods - CONNECT

The CONNECT method requests that the recipient establish a tunnel to the destination origin server identified by the request-target and, if successful, thereafter restrict its behavior to blind forwarding of packets, in both directions, until the tunnel is closed.

<https://tools.ietf.org/html/rfc7231#section-4>

# HTTP – Request Methods - OPTIONS

The OPTIONS method requests information about the communication options available for the target resource, at either the origin server or an intervening intermediary.

<https://tools.ietf.org/html/rfc7231#section-4>

# HTTP – Request Methods - TRACE

The TRACE method requests a remote, application-level loop-back of the request message.

<https://tools.ietf.org/html/rfc7231#section-4>

# HTTP - Cookies

- It's just a header, after all.
- Just keep in mind HTTP is stateless. So, cookies are used to persist state.







Her Royal  
Highness  
Xochiquezta  
“Xochi” “Lala”  
MacLalavich  
Gómez

# HTTP – Response Codes

- HTTP has a number of response codes, as well.

# HTTP – Response - Format

```
<HTTP Version> <Status Code> <Status Message>
```

```
Headers...
```

```
Body...
```

# HTTP – Response - Example

```
HTTP/1.1 200 OK ↵
```

```
Date: Mon, 08 Oct 2018 20:49:08 GMT ↵
```

```
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips ↵
```

```
X-Powered-By: PHP/7.1.10 ↵
```

```
↵
```

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
```

```
<head>
```

```
...
```

# HTTP – Response Codes

- 1xx Informational response
- 2xx Success
- 3xx Redirection
- 4xx Client errors
- 5xx Server errors

[https://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)



# HTTP – Response Codes

- The most famous one:

- **404 Not Found**

- Others:

- [https://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes#1xx\\_Informational\\_response](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#1xx_Informational_response)



# REST - Background

- **RE**presentational **S**tate **T**ransfer
- A uniform way of implementing web resources.
- Defines how to access them,
- and the responses that are allowed
- using HTTP.



# REST - Background



“‘Web resources’ were first defined on the World Wide Web as documents or files **identified by their URLs**. However, today they have a much more generic and abstract definition that encompasses **every thing or entity that can be identified, named, addressed, or handled, in any way whatsoever, on the web**. In a RESTful web service, requests made to a resource’s URI will elicit a response with a payload formatted in HTML, XML, JSON, or some other format.”

[https://en.wikipedia.org/wiki/Representational\\_state\\_transfe](https://en.wikipedia.org/wiki/Representational_state_transfe)

# REST - Background



“...there is no ‘official’ standard for RESTful Web APIs. This is because REST is an architectural style.... REST is not a standard in itself, but RESTful implementations make use of standards, such as HTTP, URI, JSON, and XML.”

[https://en.wikipedia.org/wiki/Representational\\_state\\_transfe](https://en.wikipedia.org/wiki/Representational_state_transfer)



# Caveat

- Because there is no official standard, this overview won't touch on every aspect of REST, nor is it authoritative.
- YMMV.
- Caveat Emptor.

# REST - Background

- In the beginning...
- ...HTTP was used to retrieve files from web servers.
- And that was it.

# REST – Key Concept

- Key concept: there's nothing that says that HTTP has to return an HTML file. Images are a good example.
- But even with plain (HTML) text, the Response can be anything (XML, JSON, CSV, Plain Text).

# REST – Key Concept

- Key concept: there's nothing that says that a web browser has to be a client.
- That's why I demonstrated manual connections earlier.

# REST – An Example Of A Resource That's Not A Web Page.

- RSS – Really Simple Syndication
  - XML
- Podcasts
  - Actually uses RSS

# REST

- How is it related to HTTP?

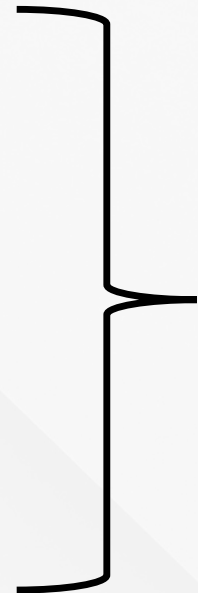


# REST – Remember These HTTP Methods?

- GET
- PUT
- PATCH
- POST
- DELETE

# REST – Remember These HTTP Methods?

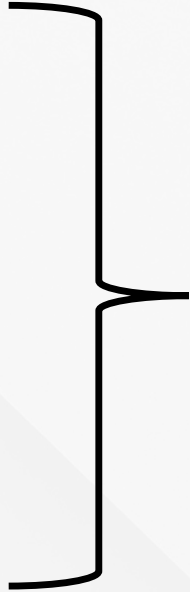
- GET
- PUT
- PATCH
- POST
- DELETE



Various actions on a resource. Basically, CRUD (Create, Read, Update and Delete).

# REST – The Methods

- GET
- PUT
- PATCH
- POST
- DELETE



These define an API  
(Application Program  
Interface)

# REST – The Methods

- GET – Retrieve a record
- PUT – Replace a record (or Create a new record)
- PATCH – Update a record
- POST – Create a new record (or Create a new Collection)
- DELETE – Delete a record

# REST – Parameters

- Can be passed as part of a URL
  - GET /myapp?id=123&format=json
  - GET /myapp/id/123/format/json



# REST – Parameters

- Or in the Body of a Request

POST <http://MyService/Person/>

Host: MyService

Content-Type: text/xml; charset=utf-8

Content-Length: 123

<?xml version="1.0" encoding="utf-8"?>

<Person>

<ID>1</ID>

<Name>Gregorio</Name>

<Email>gogogo@unm.edu</Email>

</Person>

# REST – Content-Type Header

- A REST service should include a Content-Type header in its response
- Content-Type is used to help the consumer know what kind of content to expect.

# REST – Content-Type Header

- Examples:
  - application/json
  - text/html
  - text/plain
  - application/xml
  - text/csv
  - text/css
  - application/msword

# REST – Accept Header

- The client can also ask for a particular type
- Uses the same codes as Content-Type.

# REST – Content Negotiation

- The client could also include something in the request
  - URL: `someservice.com/type/xml`; `someservice.com?type=json`
- Of course this all depends upon what the application is capable of.
- Did I mention that REST isn't a standard?



# REST – Creating REST APIs

- These days, writing REST-based web services is pretty easy.
- Most frameworks support REST
- Laravel (php) favors REST for all of its transactions
- Laravel fakes PUT, PATCH, and DELETE with hidden form fields.
- YMMV.



# REST - Demonstration

- This time, I'll use HTTPie.
- HTTPie:
  - A command line HTTP client.

# REST – Security

- Out of scope for this presentation, but...
- Use HTTPS.
- Don't pass things in the URL string.
- Use API keys
- Restrict API methods
- Don't trust parameters (validate all data)
- Use a framework

# REST – Other Issues

- Same Origin Policy
- CORS



# HTTP & REST – Other Issues

- Despite being simple, HTTP and REST are surprisingly robust and subtle.
- YMMV



Greg Gómez  
Web Developer  
UNM IT  
gogogo@unm.edu



# INFORMATION TECHNOLOGIES

**THANK YOU!**  
QUESTIONS?

**FOLLOW US**

 [fb.me/it.unm](https://fb.me/it.unm)  [unm\\_it](https://www.instagram.com/unm_it)  [@unm\\_it](https://twitter.com/@unm_it)  [@unm\\_cio](https://twitter.com/@unm_cio)  [@unm\\_it\\_alerts](https://twitter.com/@unm_it_alerts)

<https://thecatapi.com/>

# Op Cit (Incomplete)

- <https://www.restapitutorial.com/httpstatuscodes.html>
- [https://en.wikipedia.org/wiki/Representational\\_state\\_transfer](https://en.wikipedia.org/wiki/Representational_state_transfer)
- <https://restfulapi.net/>
- <https://www.restapitutorial.com/lessons/restquicktips.html>
- <https://thecatapi.com/>
- <https://quotes.rest/#/qod>
- [https://www.owasp.org/index.php/REST\\_Security\\_Cheat\\_Sheet](https://www.owasp.org/index.php/REST_Security_Cheat_Sheet)
- <https://httpie.org/>

# REST

- Template....