RESTful Web Services and conventions

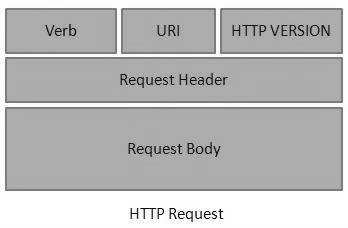
* REST (Representational State Transfer)
  + State refers to the resources you send back
  + You are sending back the state of that resource
  + State the data/fields associated with that resource
* HTTP Only
* Constraints
  + Client-server architecture
    - Separation between the back-end logic and server and the client who is using this REST service
    - This should apply to ALL services not just REST
  + Cacheable
    - You can \*choose\* to cache information on either the front end or the back end to expedite requests
  + Stateless
    - No session handling in a RESTful web service
    - Feeds into requests must be self- descriptive and contain all information necessary to process it
    - No information on the client request is stored on the backend
  + Layered System
    - REST web services should be able to stack on top of one another
    - One rest service can call another and so on
  + Code on demand (optional)
    - REST services can return code to the client
    - This code could be used to execute and perform actions
  + Uniform interface
    - The naming conventions for URI’s and methods available at URIs
      * Identification of resources through uri
      * Manipulation of Resources through representation
        + A resource is available as JSON, pdf, XML etc
      * Self descriptive messages
        + Every request contains all the information necessary to process it
        + No request is dependent on another request for it to be successful
      * HyperMedia As The Engine Of The Application State
        + Responses from a rest URI should return links to other resources on the API
        + Designed so that you as a developer and consumer of the API can try to find related URIs

Security in REST application

* REST applications users cannot login and create a session
  + REST is sessionless
  + You do not maintain essions
* When a “login” request is made you send the user an API key or a token
  + This token is then passed into requests authorizing that person to access whatever resource they need to
  + This token is usually in the header of requests

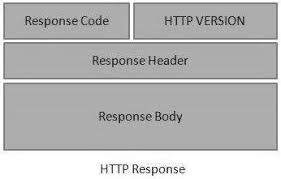
HTTP Request

* Headers
  + Key value pairs that contain meta information for the request
* HTTP version number
* HTTP verb
  + Get, put, post, delete
* Body
  + This is going to contain the content of an http request



HTTP Response

* Headers
* HTTP version
* BODY
* RESPONSE CODE
  + 100’s
  + 200’s
  + 300’s
  + 400;s
  + 500’s



When you get response back the headers arrive first.

THEN the body arrives