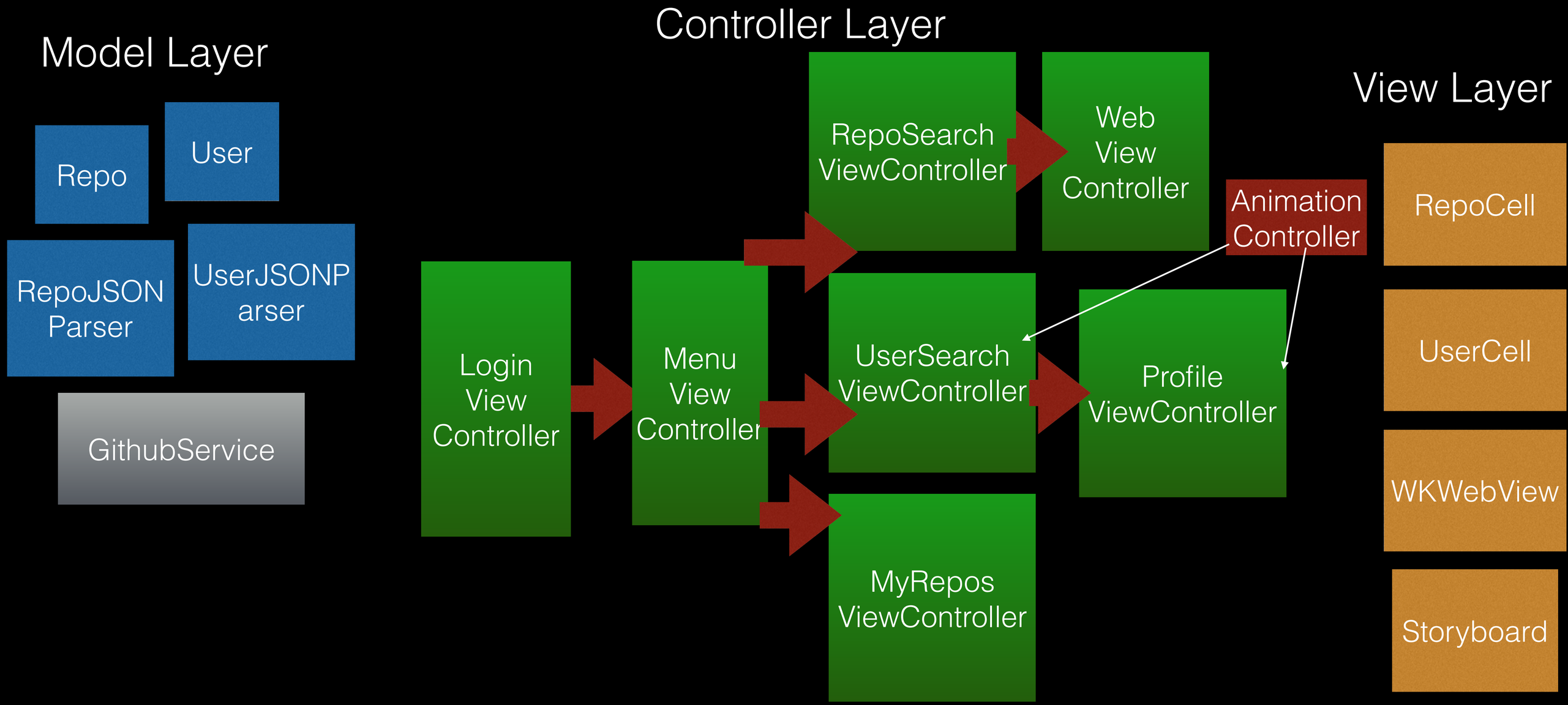
iOS Dev Accelerator
Week3 Day1

- Static Table View
- HTTP & Rest API
- NSURLSession
- UISearchBar

The MVC layout of our Week 3 App



Static Table View

Static Table Views

- Static Table Views were a new feature that debuted with storyboards in iOS 5
- Cant do them in Nibs
- Instead of the table view dynamically laying out the cells by querying the datasource, the table view relies strictly on the layout (and number) of cells on the storyboard
- **Must be laid out in a UITableViewController or subclass of it**

Why Static?

- Static table view are great for creating menus or settings screens
- Specifically when the number of rows does not have to be dynamic or change over time
- Way easier to create distinct cells vs all the cells looking the same in a dynamic table view
- Obviously, way less code to write. The best line of code is the line you don't write (this is an original quote authored by me)

Demo

HTTP & Client/Server Model

TCP/IP

- There is a computer networking model called the Internet Protocol Suite, or sometimes referred to as **TCP/IP**
- It contains a large number of protocols used by computer networks (like the internet) to specify how data should be packetized, addressed, transmitted, routed, and received at the destination
- TCP/IP is organized in four layers of abstraction..

The 4 layers of TCP/IP

- Application Layer: **HTTP**, FTP, SMTP, IMAP, SSH, Telnet, and more
- Transport Layer: TCP, UDP, DCCP, SCTP, RSVP, and more
- Internet Layer: IP (v4 and v6), ICMP, IGMP, IPsec, and more
- Link Layer: ARP, NDP, MAC, PPP, and more

HTTP (Hyper Text Transfer Protocol)

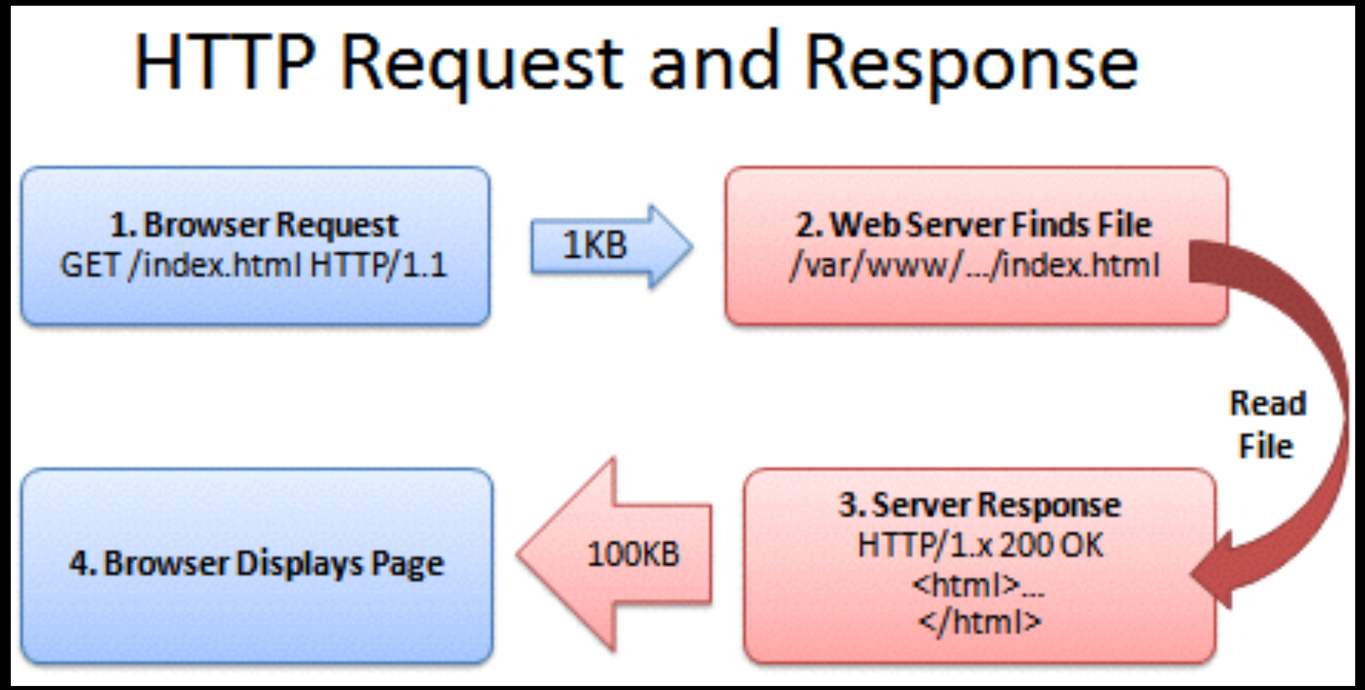
- **HTTP** is an application protocol that computers on the internet use to format and transmit resources (pages,files, data) on the web.
- It defines how web servers and clients should respond to various commands.
- “HTTP is the foundation of data communication for the World Wide Web” -Wikipedia (they did it again!!!!)

HTTP & URL

- A **resource** is anything that can be identified by a URL. It is the R in URL.
- So your browser is considered an *HTTP Client*, and it sends requests to *HTTP(s) Servers*.
- The default port for HTTP servers to listen on is port 80, but they can use any port.
- It is a stateless protocol, because each request does not know about any of the previous requests.

HTTP & Client-Server

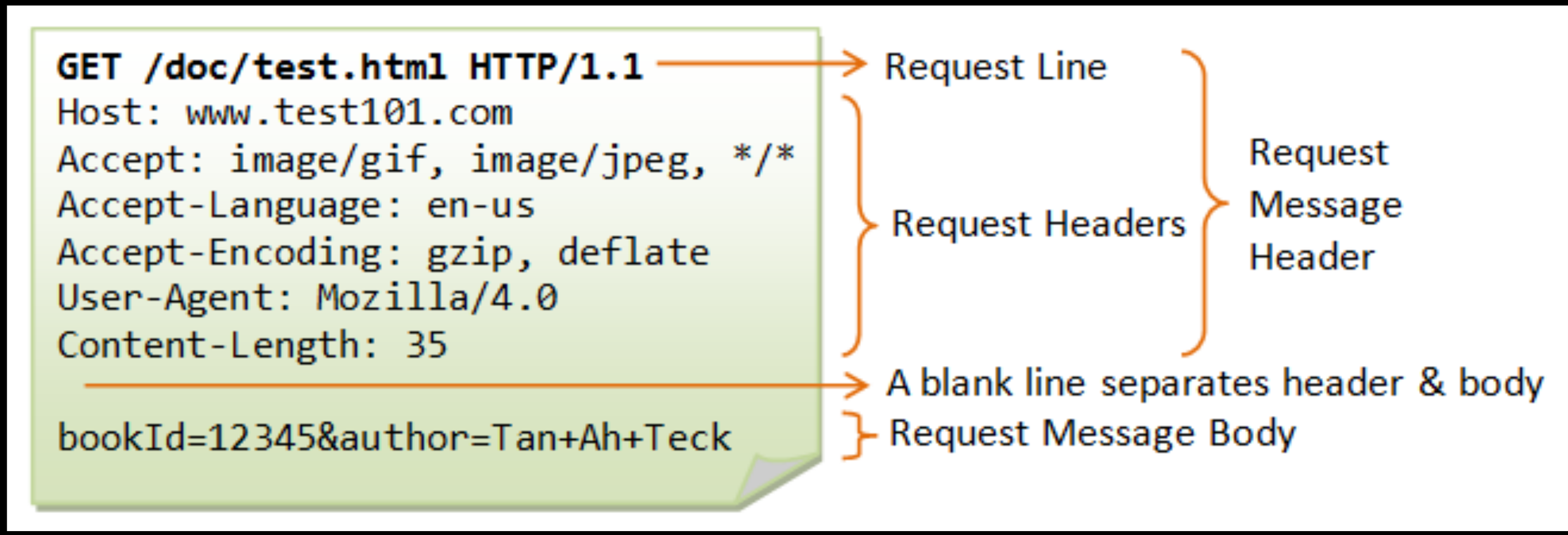
- HTTP uses a client-server model.
- A client sends a request to a server, and then the server returns a response message, which most often will contain the resource that was requested.



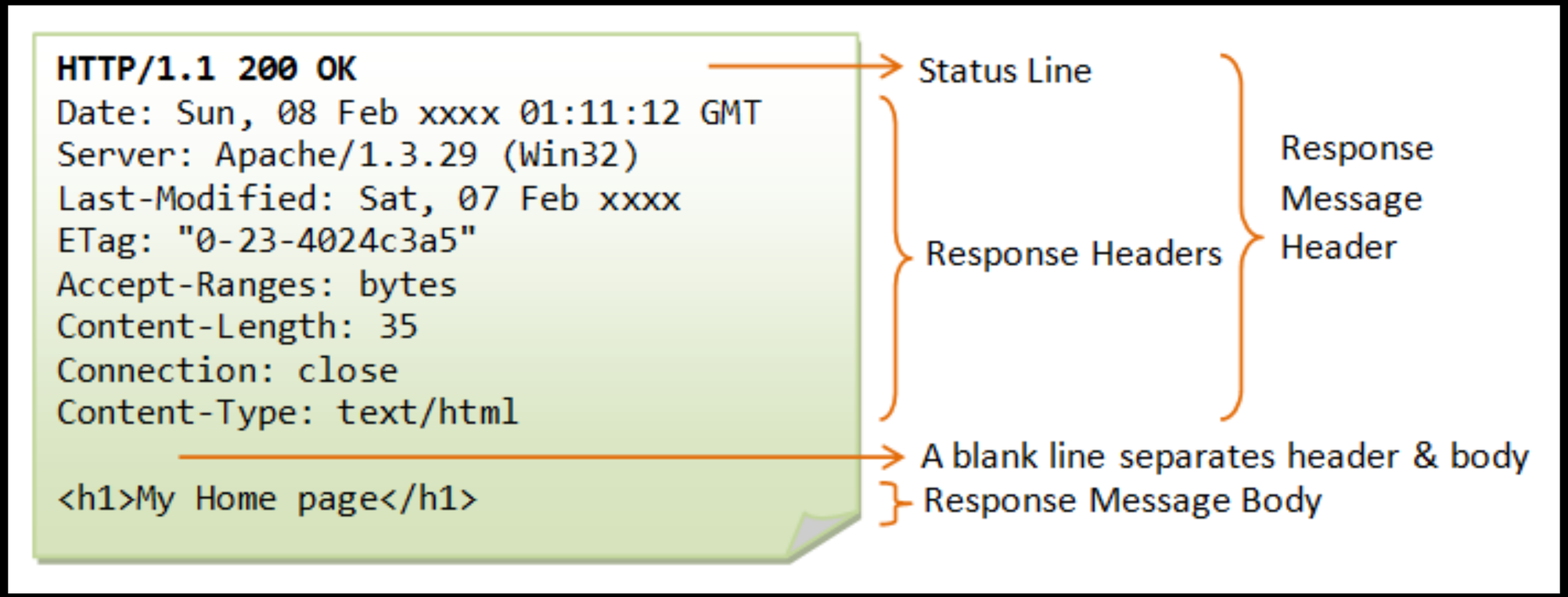
Request and Response Format

- The format of a request and response are very similar.
- They are considered ‘English-oriented’ and human readable.
- Both start off with an initial line.
- The initial line is where the main differences are between requests and responses.
- After the initial line, there can be zero or more header lines.
- After the header line is a blank line
- And finally you have the optional message body.

Request Example



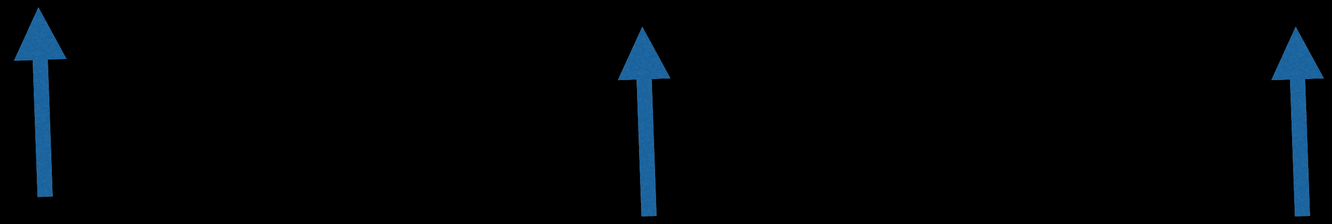
Response Example



Request Initial Line

- A request line has three parts and they are **separated by spaces**.
- The first part is the method name (more on this in a bit)
- The second part is the local path of the request resource
- and the last part is the version of HTTP being used
- Example:

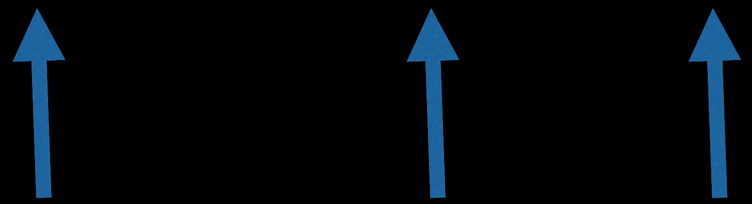
GET /NFL/seahawks/tickets.html HTTP/1.1



Response Initial Line, aka the Status Line

- A status line has three parts and they are **separated by spaces**.
- The first part is the version of HTTP being used
- The second part is a response status code
- and the last part is an english reason phrase describing the status
- Example:

HTTP/1.1 404 Not Found



Most common HTTP Methods (Verbs)

- GET - Most common HTTP method. Retrieves whatever resource is at the URL location. Your browser history is just a history of all the GET requested you have made.
- POST - Method used to request that the server accept data enclosed in the request. When you tweet, you are using POST to create a new record.
- DELETE - Used to delete a resource.
- PUT - Similar to POST, but instead of creating a new record, you are updating a preexisting one.

HTTP Status Codes

- For the most part only a few status codes that mobile apps need to check for
- 200 OK - standard response for a successful HTTP request
- 400 Bad - bad request, most likely syntax error
- 401 Unauthorized - authentication was required but not provided or incorrect in request
- 403 Forbidden - Request was valid, but the server refuses to respond.
- 404 Not found - The requested resource was not found
- 429 Too Many Requests - rate limited
- 5xx Server Error - not your app’s fault!

Header lines

- Header lines are one line per header and they take the format of:
  - “Header-Name: value”
- The header name is not case-sensitive
- You can have as many spaces or tabs between the : and the value
- Header lines that begin with space or tab are a part of the last header line for easy multi-line reading.

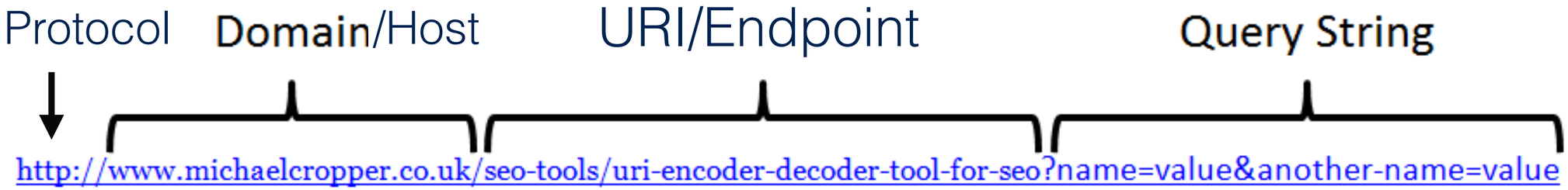
Specific Headers

- While the HTTP protocol itself does not require any header fields to function, API services require certain header fields to be set. Good API's will tell you exactly which header fields you need
- Content-Type header field is used to specify the nature of data in the body of the request. **You usually need to manually set this if you are doing a POST/PUT call**
- Authorization header field can be used to pass credentials for protected resources. Usually an OAuth token (OAuth tomorrow)
- The Apple provided classes we use to make network calls will fill out most of the headers we need to make our requests (yay)

The Message Body

- Any HTTP message may have a body after the header lines.
- In a request, this is where the appropriate data or files are placed in a POST.
- In a response, this is where the requested resource is (HTML, JSON, XML, etc)
- Whenever there is a body, Content-Type and Content-Length are usually included in the header lines so the client cant make sure everything came over the wire as intended

URL



Domain: The physical server where your website is hosted
URI: The identifier which maps to files on your server
Query String: Part of a GET request to easily pass in values to customise the output

* **Note:** URI stands for Uniform Resource Identifier

HTTP GET Example

- Lets say I wanted to get the box score for a specific mariners game. Heres the URL I would use for my GET request:
  - http://scores.espn.go.com/mlb/boxscore?gameId=340718103
- The protocol is http://
- the domain/host is scores.espn.go.com
- the endpoint/URI is mlb/boxscore
- the query string is gameId=340718103. The beginning of the query string is always marked by a question mark. You can add multiple parameters by appending them with a & (gameID=43&playerID=24543&playID=334252354)

Demo

Web APIs

- Now that you Learned The Internet, lets look at how our iOS apps can communicate with web apps/sites
- API, or Application programming Interface, is way for parts of software to interface with other software.
- Web APIs, what you will be working with, are defined as a set of HTTP requests and response messages usually in JSON or occasionally in XML.
- Most apps on your iPhone are just clients for a web api (Facebook, Twitter, Instagram, Spotify, etc)

REST APIs

- REST is an acronym for **RE**presentational **S**tate **T**ransfer. It is not a protocol, it is just a architectural style.
- You will see the term REST API a lot during your job search. It's a bit of a buzz word now, but it is essentially an API that follows certain constraints:
- Uniform Interface: Resourced based Endpoints that are consistent
- Stateless: The required state to handle the request is all contained within in the request itself. The server doesn't need to keep track of communication histories.

REST APIs

- Cacheable: Responses must defines themselves as cacheable or not
- Client-Server: Client and Server concerns are separated
- Layered System: Client doesn’t know(or care) if they are connected to the main server or a load balancing server.
- Code on Demand (optional): Servers can temporarily extend or customize functionality of a client by transferring logic to them.

Web API Client workflow

1. Clients makes a request to the server at a specific endpoint
2. Server receives request, does some server magic (queries DB/ generates data), and then sends back response
3. Client checks the response http status code
4. If its in the 200 range (everything is good), client parses the JSON response into model objects
5. Client updates UI and state with newly acquired model objects

NSURLSession

NSURLSession

- The NSURLSession class and related classes provide an API for making HTTP requests.
- NSURLSession is highly asynchronous.
- NSURLSession has two ways to use it, with completion handlers (closures) or delegation.
- We will focus on the closure aka callback way

NSURLSession Setup

- NSURLSession is initialized with a NSURLSessionConfiguration.
- NSURLSessionConfiguration has 3 types:
    1. Default session - disk based cache
    2. Ephemeral session - no disk based caching, everything in memory only
    3. Background session - similar to default, except a separate process handles all data transfers
- Configurations have many properties for customization. For example you can set the maximum number of connections per host, timeout intervals, cellular access or wifi only, and http header fields.
- NSURLSession also has a sharedSession singleton available to use based on a default session. We will use this one.

NSURLSession Setup

```
  var configuration =
NSURLSessionConfiguration.ephemeralSessionConfigur
ation()

    self.urlSession = NSURLSession(configuration:
configuration)
```

NSURLSession

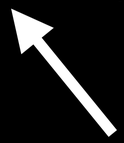
- All http requests made with NSURLSession are considered ‘Tasks’. Think of them as little minions for your session
- A task must run on a session (NSURLSession class)
- Three types of tasks:
    1. Data tasks - receive and send data using NSData objects in memory
    2. Upload tasks - send files w/ background support
    3. Download tasks - download w/ background support

NSURLSession Data Task

```
var request = NSMutableURLRequest(URL: NSURL(string: "https://api.github.com/search/repositories?q=\
(string)"))
        request.HTTPMethod = "GET"

        let repoDataTask = self.urlSession.dataTaskWithRequest(request, completionHandler: {(data,
response, error) in

            if error {
                //do something for general error
            }
          else {
          //do switch statement on response code and do something with the data
          }
            })
        repoDataTask.resume()
```



Don’t forget this line!!!

Demo

UISearchBar

UISearchBar

- The UISearchBar class is a text field based control.
- The search bar provides a text field for text input, a search button, a bookmark button, and a cancel button.
- It relies on it’s delegate to perform searches when one of its buttons is pressed.
- You can also embed a search bar into a tableview by dragging it into the tableview on storyboard.

Demo