
# HTTP and 3rd Party Libraries

Dr. Sarah Abraham

*University of Texas at Austin*
*CS329e*
*Spring 2020*

# HTTP

- Hypertext Transfer Protocol
  - Nodes contain hypertext
  - Hyperlinks connect between nodes
  - HTTP allows for the exchange and transfer of hypertext
- HTTP requests submitted to the server
- Server responds with message information and requested content
- HTML is hypertext markup language

# Messages

- HTTP messages consists of:
  - Initial line
    - Request: `METHOD pathToRequestedURI HTTP/x.x`
    - Response: `HTTP/x.x status_code reason`
  - Header information
    - Header-Name: value
    - https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html
  - Message body

# Status Codes

- Sent in initial line of responses
- Indicate success or failure of request
- Common status codes:
  - 200 (ok)
  - 404 (not found)
  - 301 (moved permanently)
  - 500 (server error)

# Other Status Codes

- 1xx (request received, continuing to process)
- 2xx (request received, understood, accepted and processed)
- 3xx (client must take additional action to complete request)
- 4xx (client error in request)
- 5xx (server error in fulfilling request)

# GET and HEAD Methods

- Retrieves information from given server at specified URI
- Does not modify data in any way
- GET retrieves headers and message body
- HEAD retrieves only the headers

# POST and PUT Methods

- Sends data to server
- Enclosed entity (block of data) sent with request
- POST method submits data to be processed by resource at URI
  - URI is a program for handing this data
  - Response is program's output
- PUT method submits data to be stored at URI

# DELETE

- Removes all representations of the resource identified by URI

- Should flag cached representations of resource as stale

  - That is, *all* representations — even those in cache — should be removed from the client's view of the server

- Server can implement by deleting data or moving data to an inaccessible location

# Quiz Question!

- True or False: The POST and PUT method both send data to a server.

# REST

- REpresentational State Transfer
- Architecture and practices to provide interoperability between computers across the Internet
- Compliant systems access and manipulate web resources using predefined operations
- Responses provided via XML, HTML or JSON
- “Best practices” for using HTTP

# REST Properties

- Performance
- Scalability
- Simplicity
- Modifiability
- Visibility
- Portability
- Reliability

# REST Constraints

- Uniform Interface
- Stateless
- Cacheable
- Client-Server
- Layered System
- Code on Demand (optional)

# RESTful APIs

- Web service APIs that adhere to REST constraints
- Must have:
  - A base URL
  - Internet media type (MIME type)
  - Standard HTTP methods

# AFNetworking/Alamofire

- Built on top of `URLSession` and Foundation's URL Loading System
- Allows for easier HTTP requests and response handling without all the boilerplate code
- Example:

```
let request = Alamofire.request(.GET, "http://
httpbin.org/get")

   request.validate() //Checks response status code

   request.response { (request, response, data, error) in

   }
```

# Importing Alamofire

- Xcode seamlessly handles importing built-in libraries
  - You have been calling `import UIKit` at the top of every ViewController to import UIViewController functionality
  - If a custom class is included in a project, Xcode automatically can link it into another class
- Alamofire is not built into Apple's standard libraries
  - Must import it as its own library

# Cocoapods

- *Dependency manager* for Swift projects
- Program knows about many libraries built for use in Swift/Objective-C
  - Allows easy downloading of libraries
  - Allows easy connecting of libraries to a project
- Standard way of managing libraries for Swift

# Installing Cocoapods

- Generally installed and use via command line
  - There is an app (https://cocoapods.org/app) but frequently tools management is done from command line anyway…
    - Command line is found under: Applications -> Utilities -> Terminal
- From terminal call `sudo gem install cocoapods`
  - Built on Ruby (will work without additional installs on OSX)
  - May have to provide user password

# Using Cocoapods

- From terminal, can call commands using `pod`
  - `pod init` creates a file (Podfile) that manages libraries for the project (call from within project folder)
- Must manually add frameworks to the Podfile:

```
platform :ios, '12.0'
use_frameworks!
target 'MyApp' do
 pod 'Alamofire', ‘5.0.0' #specifies version
end
```

# Installing Dependencies

- Once Podfile exists in project, must install all requested frameworks before they will connect to the project
  - call `pod install` from within the project directory
- This will create a `.xcworkspace` file
  - *Must* access project using `.xcworkspace` rather than `.xcodeproj` from now on
  - Frameworks are not part of your project — workspace include your work and other files you need

# Workspace versus Project

## xcodeproj space