Web App Architectures

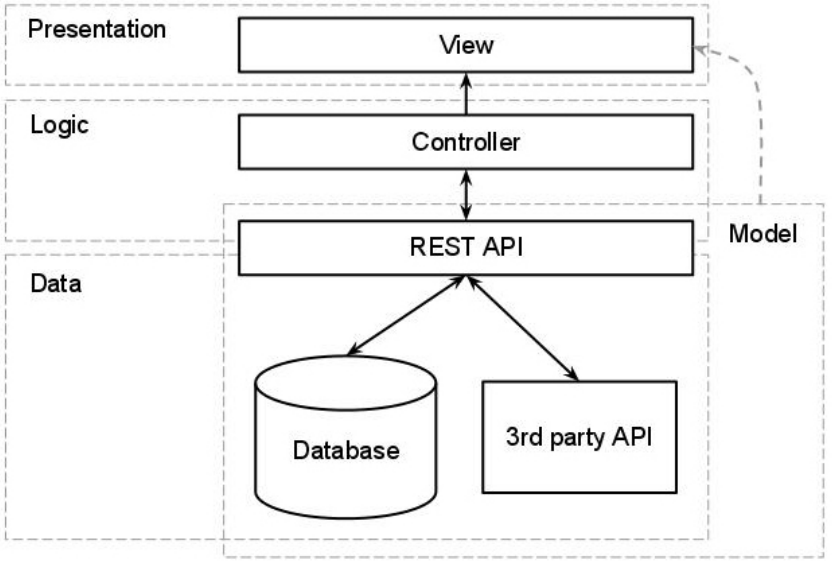
FROM FRONT END TO BACK END

What is a web app?

- A web application can be described as a server-client software application where the user interface runs in a web browser.
- A web application can be built using a variety of different “stacks” and “architectures”.
- A “stack” is a set of common technologies used to deliver your web application.
- A ”architecture” is a set of application design principles meant to ensure stability, scalability, reliability and manageability. There are many different architectures, in this presentation we are going to focus on the most common ones in the job market.

Basic Terminology

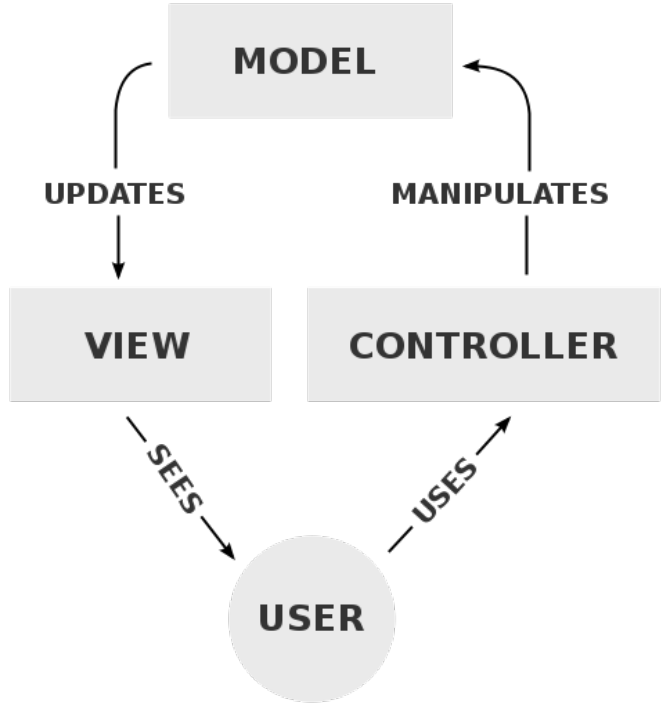
- The client is your users browser, i.e. firefox, chrome, etc etc
  - The client caches html, css and javascript onto your users local file storage. (remember when you clear the cache, some sites take longer to load)
  - The client communicates with the server using http protocols, i.e. fetch requests
- The server is what "serves" your application stuffs.
  - The server usually contains your business logic, but not always.
  - The server is where your application does it's super awesome magic.
  - The server "serves" the client html, css, js and data upon an http request.



MVC- Model View Controller

This is arguably the MOST COMMON FRAMEWORK, you MUST be able to understand how it functions. Traditionally a "server-side" framework: meaning almost all of the data or files must be requested from the server, which requires a "trip" to the server.

- Examples of MVC stacks would be:
  - LAMP- linux(os), apache(server), mysql (database), php (programming language
  - (windows stack)/ASP.net Stack – SQL Server, mysql (database), C# (programming language), ASP.net (framework)

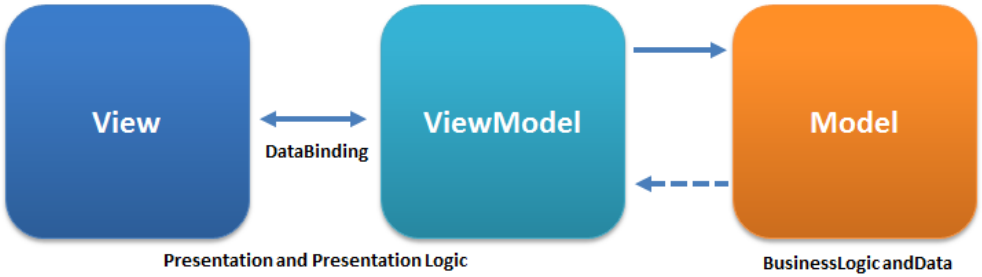


MVC - The good and the bad

- The goods:
  - LOTS of documentation
  - Pluggable views and controllers (usually)
  - Easy to change user interface without changing backend.
- The cons:
  - Very complex
  - Leans towards being very monolithic
  - Easy to do wrong very difficult to do great
  - Every new view requires additional trip to server

MVVM- Model, View, View-Model

- Usually the rendering is done "client side", which means the html (views), css, and JS all gets preloaded, and is available in cache. No server side response is needed to change the view, but this is not a requirement as MVVM softwares do not have a traditional"client".
- This pattern usually involved an API on the backend which serves data to the front end to be manipulated and displayed.
- Example MVVM stack:
  - AngularJS SPA: Server (server agnostic), Database (database agnostic), AngularJs



MVVM- The good and the bad

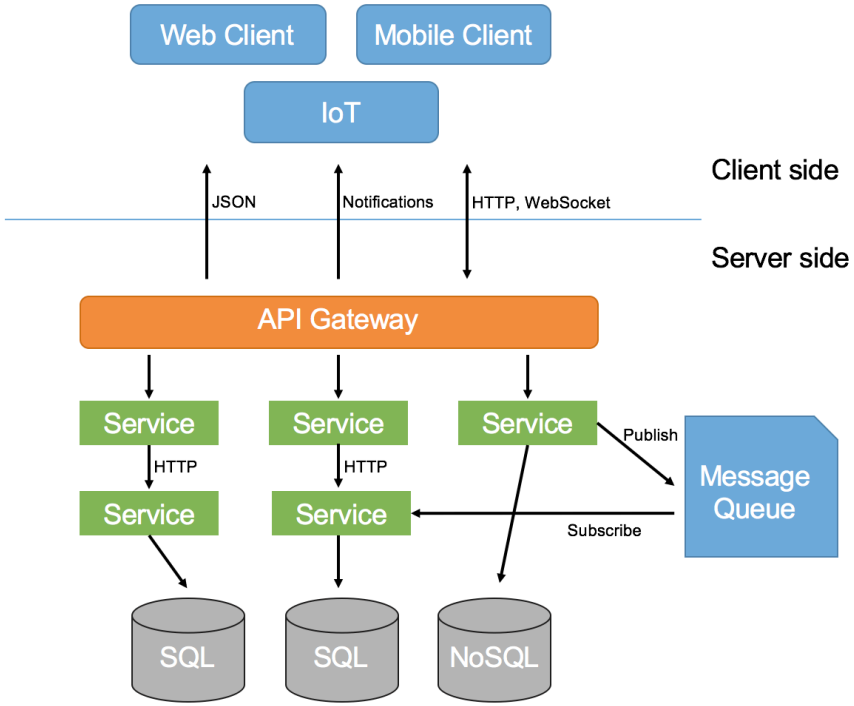
- Pros:
  - Promotes loose coupling (models are by request, not tied to controller)
  - Promotes maintainability (Backend Agnostic, easier for front end devs)
  - Uses data binding, which allows for increased testability
  - With an api in place, moving to a mobile app is easier.
- Cons:
  - Initial page load is usually high
  - Slightly more effort to build, more thought goes into it.

SOA – Service Oriented Architecture

Services are comprised of a service provider and a service consumer. The service provider is some method or function that does a specific thing. I like to simplify them as a the smallest "unit of work" that needs to be done.

- 'Find Cars' could be a service but an entire 'cars class' (all methods, finding value, adding taxes, adjusting inventory) would not be a service.

Services are meant to be highly modular, meant to be easily changed and very loosely coupled. This makes them very useful for communicating between different parts of your stack.



SOA – The good and the bad

- Pros:
  - Highly modular
  - Loosely coupled, less refactoring
  - Easy to get new people up to speed/productive
- Cons:
  - Dev time to set up is normally higher because of all the abstraction
  - Services have to be "available" or application may not work as intended
  - Although easy to work with 1 service, juggling 60+ takes talent (a la netflix)

Resources

- SOA/Microservices presentation: https://www.youtube.com/watch?v=B8E45FZ9pko
- MVC: https://www.youtube.com/watch?v=1IsL6g2ixak
- MVVM/AngularJS: https://www.youtube.com/watch?v=ejBkOjEG6F0