CS 169 Discussion 2

Administrivia

HW1 Ruby Intro is due this Friday 9/13

Make sure to fill out team matching form by tonight!

Access discussion slides, worksheets, and solutions at **srujayk.com/cs169**

Office Hours: Monday 2-3 in Soda 341B (Undergrad Lounge)

Agenda

- SaaS Architecture
- Service Oriented Architecture
- APIs
- RESTful Thinking
- URIs
- Worksheet
- Intro to Sinatra

Saas Architecture

Overview

- **One** server (software) to **many** clients
  - Berkeley provides education to many students
  - Facebook, Netflix, Bloomberg, etc.
- Communication through HTTP protocol
  - One of many client-server, request-reply transport protocols
  - HTTP is **stateless**

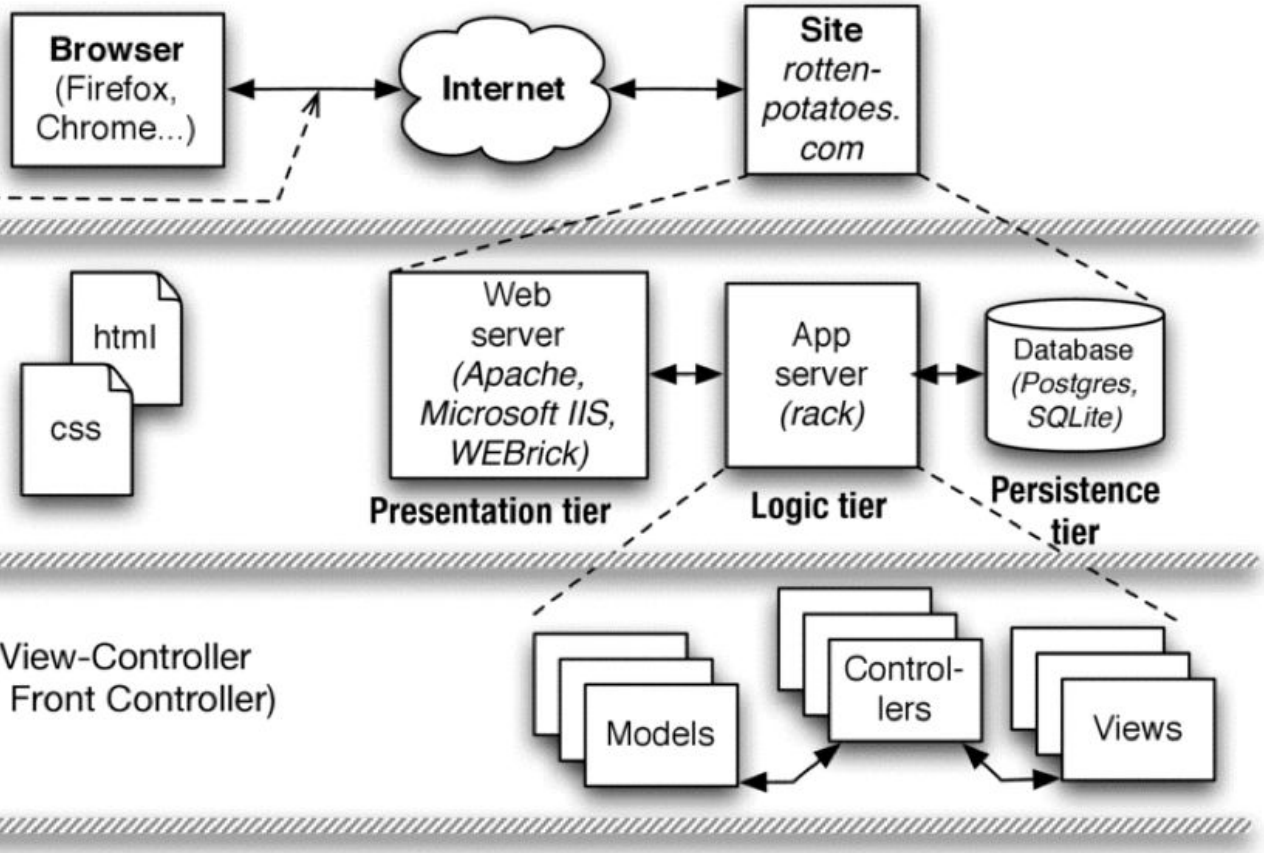
§2.1 100,000 feet
- Client-server (vs. P2P)

§2.2 50,000 feet
- HTTP & URIs

§2.3 10,000 feet
- XHTML & CSS

§2.4 5,000 feet
- 3-tier architecture
- Horizontal scaling

§2.5 1,000 feet—Model-View-Controller (vs. Page Controller, Front Controller)



§2.6 500 feet: Active Record models (vs. Data Mapper)
§2.7 500 feet: RESTful controllers (Representational State Transfer for self-contained actions)
§2.8 500 feet: Template View (vs. Transform View)

- **Active Record**
- Data Mapper
- **REST**
- **Template View**
- Transform View

SOA

Service Oriented Architecture (SOA)

- Service is the fundamental building block of a software system
- A service is a program that can be interacted with through a well-defined set of message exchanges
  - Typically encapsulates a high level business concept
  - Communicate with one another through APIs

APIs

Application Programming Interfaces (APIs)

- An API is a set of well-defined methods for interacting with the data of a software system
- Different levels of APIs
  - Libraries and frameworks, OS level APIs, etc.
  - We'll be focusing on web APIs

RESTful Thinking

RESTful Operations

- **Re**presentational **S**tate **T**ransfer is an architectural pattern for developing web services
- RESTful APIs are APIs that make use of HTTP for its procedures
  - GET, POST, PUT, PATCH, DELETE, etc.
- For most APIs, GET and POST are enough
- Some browsers do not support HTTP methods outside of GET and POST
  - Although they would be achievable through AJAX

RESTful APIs

Twitter

Many popular websites / online services have APIs. For example:

Top endpoints

Below are a few examples of our Twitter API endpoints, requests, and responses. These examples use twurl—a command-line application that can be used to make authenticated requests to the Twitter platform. *twurl is like curl*, except that it abstracts away OAuth details once you configure it with your keys.

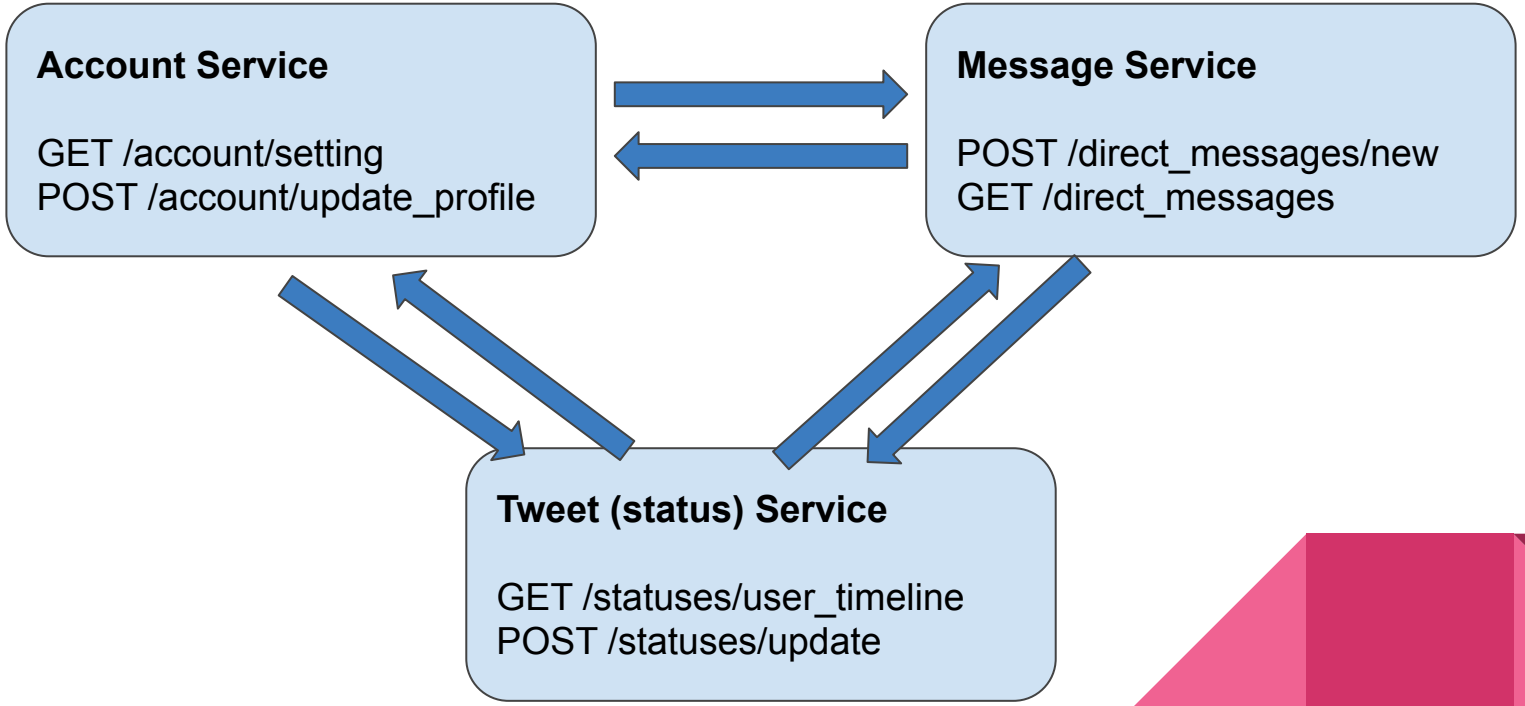
Search API | Ads API | Engagement API | Direct Message API | Account Activity API | Embed a Tweet

Search Tweets published in the last 7 days.

```
# Request Tweets from last 7 days
# Search query: from:Nasa OR #nasa

twurl "/1.1/search/tweets.json?q=from%3ANasa%20OR%20%23nasa"

# Response, an array of Tweet JSON:

{
  "statuses" : [
```

Documentation ›

Twitter in SOA example



URIs

Uniform Resource Identifiers (URIs)

What’s in a URI? Take https://github.com:443/cycomachead?tab=repositories.

- https:// → protocol; others include ftp, smtp
- github.com → host
- 443 → port to connect to on destination server
- /cychomachead → relative path on server
- ?tab=repositories → query parameters

URIs and RESTful Conventions

- If your API works with various resources, like **users** and **books**, then your routes will be structured like so:
  - /users/create
  - /user/117/edit
  - /user/117/books
  - /user/117/book/18
- In general, /<resource>/<property or subresource>

URIs and RESTful Conventions

GET /discussions/2/presentation

GET /discussions/2/worksheet

POST /discussions/2/worksheet

POST/discussions

GET /getDiscussion

GET /getDiscussionById/2

POST /createDiscussion

POST /worksheetForDiscussion/2

Let’s design one together

Goal: Build an authentication server

What APIs do we need?

Display Login Template?

Send Credentials?

Need to go find my credentials?

Need to update my credentials?

Let’s design one together

Goal: Build an authentication server

What APIs do we need?

Display Login Template? GET /login

Send Credentials? POST /login

Need to go find my credentials? GET /login/update

Need to update my credentials? POST /login/update

Activity: Design **Venmo** as SOA

Venmo? Online platform where users can send money to each other.

Objectives: Design functional services and expose necessary APIs for each service.

Requirements

a. Users are able to log in and see their profiles
b. Users can transfer their balance to their checking/saving account
c. Users can make transactions (making payments or requesting payments)
d. Users can see the list of transactions
e. Anything you feel necessary. Explain why.

Example Solution

Authentication Service

GET /login
POST /login
POST /login/update

Banking Service

POST /charge_debit
POST /transfer
POST /charge_credit

Account Service

GET /account/profile
GET /account/balance
POST /account/update
GET /account/search

Transaction Service

GET /transaction/stream
GET /transaction/list
POST /transaction/send
POST /transaction/request
POST /transaction/fulfill

Worksheet

Intro to Sinatra

Sinatra

- Lightweight web framework that maps HTTP methods/routes to actions

Venmo SOA (Continued)

Authentication Service

GET /login
POST /login
POST /login/update

Banking Service

POST /charge_debit
POST /transfer
POST /charge_credit

```ruby
require 'sinatra'

get '/login' do
  get_login_template()
end

post '/login' do
  attempt_login(params[:password], params[:username])
end

post '/login/update' do
  edit_login(params[:password], params[:password_confirmation], params[:username])
end

post '/charge_debit' do
  user = getUserById(params[:userId])
  charge_debit(user, params[:amount])
end

post '/transfer' do
  user = getUserById(params[:userId])
  transfer_to_debit(user, params[:amount])
end

post '/charge_credit' do
  user = getUserById(params[:userId])
  charge_credit(user, params[:amount])
end
```