Web Applications: Overview and Architecture

Lecture 1
Road Map in Pictures: Web App
Road Map in Pictures

Browser → Request → HTTP → Response → Web Server

- Rails
  - Model
  - Controller
  - View
  - Dispatcher
  - Routes
  - Web Server

- Ruby
- HTML
- CSS
- JavaScript
- Rails
Road Map: Schedule of Topics

- A Language
  - Ruby
- Foundations
  - Version Control, Networking, Regular Expressions
- Static web pages
  - HTML & CSS
- Dynamic web pages
  - JavaScript
- Framework for web applications
  - Rails
- Applied Topics
  - Security, Encodings
Resources (Increasing Latency)

- Lectures, office hours, meetings
  - Instructor, grader(s)
  - Each other

- Discord Server
  - Group collaboration, messaging
  - Discussion forum, news, announcements

- Class website
  - Handouts, lecture notes, lab assignments
  - Pointers to more resources

- Carmen
  - Syllabus (note exam requirement)
  - Grades
Mens Sana in Corpore Sano

Running plan for the semester:
- Run from here to Louisville, KY
- Equivalently, run 210 miles
- Equivalently, run 8 marathons
Mens Sana in Corpore Sano

- Running plan for the semester:
  - Run from here to Louisville, KY
  - Equivalently, run 210 miles
  - Equivalently, run 8 marathons

- Alternate running plan:

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 miles</td>
<td>rest</td>
<td>3 miles</td>
<td>rest</td>
<td>4 miles</td>
<td>rest</td>
<td>3 miles</td>
</tr>
</tbody>
</table>
Technical Content

- **Languages and Technologies**
  - HTTP
  - XML, HTML, CSS, JavaScript
  - Ruby, Ruby on Rails

- **Tools and techniques**
  - Design patterns (MVC)
  - git, linux
  - Regular expressions, unicode, system time

- **Advanced topics**
  - Programming languages, networking, cryptography, databases, operating systems
Stability of Content: Concepts

- Conceptual underpinnings will be relevant forever
- In this course:
  - Single-point of control over change
  - Abstraction (vs realization)
  - Design patterns
  - Regular Expressions (the math part)
  - Cryptography (the math part)
  - Motivation for version control
  - Time-space performance trade-offs
Stability of Content: Technology

- Some technologies have been around a long time, and will likely be relevant for many more years

- Examples in this course:
  - Linux
  - SQL
  - HTTP
  - HTML
  - CSS
  - JavaScript
Stability of Content: Tools

- Some tools come and go
- They are useful for getting things done now, but may not be as relevant or fashionable in 10 years
- Examples in this course
  - Ruby
  - git
Stability of Content: Framework

- There are many frameworks and libraries for web development
- They come and go so quickly, there is always something new
- Examples:
  - Web frameworks like Rails, Express.js...
  - Ruby gems like Middleman, Nokogiri, Cucumber...
  - JavaScript libraries like Angular, React
  - HTML/CSS libraries like Bootstrap, Baseline, Foundation...
Meta Content: Software Eng.

- Lasting relevance
- Project development in the "real world"
  1. Vague open-ended requirements
  2. Large, complex problems
  3. Teams
Topic 1: Vague Requirements

- Two aspects to engineering:
  - Satisfying the constraints (solving the problem)
  - Optimizing the solution (better, faster, cheaper)
- Must first identify and understand the problem
  - Requirements elicitation
- Recognize tradeoffs
  - Improvement in one aspect at the expense of another
Topic 2: Size and Complexity

- “Programming in the large”
  - Does not all fit in one person’s head or schedule
  - Interfaces, modules, components, classes
- Design
  - Measure twice, cut once
- Process
  - Agile, waterfall, TDD,...
- Documentation
- Testing
Topic 3: Group Work

- Naïve view of CS: Lone wolf hacker
- Reality: large multidisciplinary teams
  - Developers, testers, marketing, HR, management, clients
  - Communication skills are critical
- Many challenges
  - Rely on others
  - Compromises become necessary
  - Personalities
- Many rewards
  - Accomplish more
  - Learn more
SE In This Course...

- Group work! You will be in 2 groups:
  - A “home group” for projects
  - A “technology team” for tasks
- Multidisciplinary teams
  - Tech teams cut across project groups
- Open-ended projects
- Communication skills
  - Presentations to class
Architecture: Desktop App

User Interface

Graphical events (mouse moves, button pushed)

Application

Processing, Calculating

Data

Persistence, Transactions, Triggers
Client-Server App: 2-Tier

Where should we cut?

User Interface

- Ultra-thin client (aka “dumb terminal”)
  [X11, RDP, character echo from mainframe]

- UI on client, processing just for display
  [browser rendering static HTML]

- Some processing on client
  [validate form fields before submission]

- Thick client: connect directly to DB mngr
  [native look & feel]

Application

Data

- Cache data on client
  [responsiveness, less network congestion]
Basic Web App Skeleton: 3-Tier

User Interface

http  HTML, CSS, Javascript

Application

SQL

Data
Advantages over Thick Clients

- **Performance**
  - 1 (expensive) network call to app layer results in many calls to data layer
  - Compute-intensive part on faster machine

- **Flexibility**
  - Update app logic without changing client

- **Robustness**
  - Transactions, logging at app level

- **Security**
  - Login, authentication, encryption all better at app level than data level
Web App Skeleton: 4-Tier

User Interface

Presentation Layer

Business Logic

Data
Web App Skeleton: n-Tier...

- User Interface
- Caching
- Presentation Layer
- Workflow
- Business Logic
- Data
Summary

- Technical aspects of course content
  - Many different web technologies
  - Rapidly evolving landscape
- Meta content: Software engineering
  - Vague requirements
  - Large systems
  - Teams
- 2-, 3-, 4-, n-Tier Architectures