Web Applications: Overview and Architecture

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

Browser → Request → Web Server → Response

Model → Controller → View

Dispatcher → Routes → Web Server
Road Map in Pictures

Browser → Request → HTTP → Response

Web Server

Rails

Model → View

Controller

Dispatcher → Routes

Web Server

HTML

CSS

JavaScript

Ruby
Road Map in Words

- 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

- Class website
  - Syllabus (note exam requirement)
  - Handouts, lecture notes, lab assignments
  - Pointers to more resources
- Piazza (www.piazza.com)
  - Discussion forum, news, announcements
- Carmen
  - Grades
- Instructor, TA
- Each other!
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, 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
  - Cryptography (the math part)
  - Regular Expressions (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
  - JQuery
  - 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, MooTools
  - 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
Topic 2: Size and Complexity

- “Programming in the large”
  - Doesn’t 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
In This Course...

- Group work: 4 people / team
- Multidisciplinary teams
  - I will create cross-cutting technical areas
- Open-ended projects
- Communication skills
  - Presentations to class
Architecture: Desktop App

Graphical events (mouse moves, button pushed)

Processing, Calculating

Persistence, Transactions, Triggers

User Interface

Application

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

User Interface

http

HTML, CSS, Javascript

Application

.....

SQL

Data

http, HTML, CSS, Javascript

MySQL, Microsoft SQL Server, SQLite

Apache Tomcat, Java

IIS7

PHP
Advantages over Thick Clients

- **Performance**
  - 1 (expensive) network call to app results in many calls to data
  - Compute-intensive app 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 web technologies
- Meta content: Software engineering
  - Vague requirements
  - Large systems
  - Teams
- 2-, 3-, 4-, n-Tier Architectures