Responsive Design; Mobile Development; Going Live

CPS353 Internet Programming

Simon Miner

Gordon College

Last Modified: 12/04/2013

Agenda

- Scripture (Colossians 4) and Prayer
- Check-in
- Responsive Design
- Mobile Development
- Homework 8
- Going Live: Production, Performance, and Analytics
- Milestone X

Check-in

- Milestone 8
- Web Technology Research Projects
  - Presentations next week (12/4) – 50% of credit (5% of final course grade)
  - Reports also due next week (12/4) – 50% of credit (5% of final grade)

We are in the Home Stretch!!

- Don't give up! Hang in there! We're almost done!
- Concerned about your course grade?
  - You can make 25%+ of your grade for the course over the next ~2 weeks.
    - 5% -- Homework 8 (due next Wednesday)
    - 5% -- Web Technology Research Presentation (next Wednesday)
    - 5% -- Web Technology Research Report (due next Wednesday)
    - 10% -- Milestone X (counts for double)
      - Final touches – fun, useful, less arduous features to develop
      - Not dependent on previous milestones -- You can do these regardless of where your Rails project currently stands
        - » Just need a show restaurant page and some navigation controls
      - Very little Ruby required
      - Extra credit available
      - Not due until the end of the final exam hour (12/19 at 4:30 pm)

Responsive Web Design

- Problems
  - Different browsers support different features
  - Different devices have different display sizes and shapes
- Could create custom experiences for popular browser/device combinations
  - i.e. mobile site
  - Do we also need a “tablet” site?
  - What happens when the next device comes along?
- More sustainable approach
  - Adapt site’s layout based on the viewing environment
  - Use feature detection to conditionally enable functionality if it is supported

Responsive Design Techniques

- Fluid grids – use percentages rather than absolute units to set page element sizes and positions
  - Allows the layout to automatically “reflow” for the current viewing environment
- Flexible images
- Media queries
- JavaScript Techniques

Flexible Images

- Put an image (or other media) element in a flexible container
  - Set its max-width property to 100%
  - Modern browsers will resize the image automatically as its container size changes

```
div { margin-left: auto; margin-right-auto;
width: 60%; }
img, embed, object, video { max-width: 100% }
…
<div><img src="centered-image.jpg"/></div>
```

Media Queries

- CSS @media rule allows you to define or override styles based on media type criteria
  - Can also be specified in <link> tag's media attribute or CSS @import call
- Media query tests
  - screen – Is the page being rendered on a screen (vs. print)?
  - min-width/max-width – of viewport (in pixels)
  - min-device-width/max-device-width – actual device width
  - orientation – landscape or portrait
- Can be joined together by `and` in a single statement

Media Query Example

```
#logo {
    padding: 0 10px;
    font-size: 20pt;
    font-weight: bold;
    font-style: italic;
}
@media screen and (min-width:400px) { /* larger screens */
   #logo {
    float: left;
    width: 20%
  }
}
@media screen and (max-width:399px) { /* smaller screens */
  #logo {
      margin-top: 5px;
      text-align: center;
  }
}
```

Media Query Guidelines

- Design media queries based on your application's content
  - Not particular device sizes. (These will change.)
- Making tables and form elements responsive can be a bit trickier
- Resources
  - http://mediaquer.ies -- Collection of good examples.
  - Firefox's Responsive Design View (and/or similar browser features and extensions)

JavaScript and Responsive Design

- CSS excels at making elements or portions of a layout responsive
- If behavior or feature sets need to change for different sized layouts, JavaScript will likely be needed
  - Progressive enhancement – check for required functionality before enabling features
  - Graceful degradation – ensure the application still works as much as possible for clients missing key pieces of functionality

Mobile Development

- Developing for smaller untethered devices has significant ramifications
  - Smaller screens mean less screen “real estate” for content
  - Touch and gesture based interactions
    - Tapping, swiping, pinching…
  - Resource management
    - Memory/storage limitations
    - Network latency
    - Limited battery life
- Approaches
  - Mobile web
  - Native applications

Mobile Web

- Takes advantage of newer HTML5 and CSS3 features
  - More consistently available on mobile browsers
  - Local storage, geolocation, CSS3 columns
- The browser is the platform
  - Standardizes user experience across devices
  - Enables continuity between data and UI between devices (desktops, netbooks, tablets, phones)
    - Makes code and logic reusable
    - Example – Christianbook.com mobile site
- Limited feature set
  - Often cannot take advantage of devices' full power and potential
  - Need to find and support the least common denominator in mobile browser feature set
    - Can lead to less engaging user interfaces and experiences

Native Applications

- Written using SDKs specific to the device or mobile OS
- Allow applications to take advantage of device features and resources
  - Client data
  - Inter-application communication
  - Specialized software libraries, toolkits, and widgets
  - Performance benefits not available in a browser environment
  - Enable more polished and intuitive user experiences
- Make working across platforms / applications more challenging
  - Web services are the primary conduit for shared data and logic
- Device / OS vendor owns and manages the platform.

Android

- Google's mobile operating system
  - Releases are the named after desserts

    Jellybean, Ice cream sandwich, KitKat, Froyo
  - Primary development environment
    - Java, ADT (Android Development Toolkit, and Eclipse (or some other IDE with appropriate plugins)
- Open(ish) ecosystem
  - Lots of devices – Droid, Nexus, Kindle
  - Anyone can create and submit an application to the Android Marketplace
  - Android application package (APK) files can be migrated and "side-loaded" onto various devices
    - Example: CBD Reader on Kindle
- Applications built using MVC
  - Controller is replaced by the idea of an "Activity" – a single focused thing that a user can do

iOS

- Apple's proprietary OS that runs on the iPhone and iPad
- Applications typically written in Objective-C on Xcode
  - Built using MVC – ViewControllers, Core Data, Core Location, Cocoa Touch
- Apple distributes iOS apps through its App Store
  - Carefully reviews and filters what gets into the store
    - All new and updated apps need to go through a multi-day/week approval process
  - Maintains a "walled garden" with strict rules, limitations, and "preferences" for development practices
    - I.e. Adobe Flash does not work in iOS

Homework 8

Going Live

- Software environments
  - Development
  - Test
  - Staging
  - Production
- Production environment characteristics
  - Fast
  - Scalable
  - Reliable
  - Secure
  - Closely monitored

Rails in Production

- Upgrading the web server
  - From WEBrick
  - To Apache, nginx, lighttd, etc.
- Upgrading the database
  - From SQLite3
  - To MySQL, Oracle, MariaDB, etc.
- Adding an application tier
  - Passenger

Rails Deployment Example

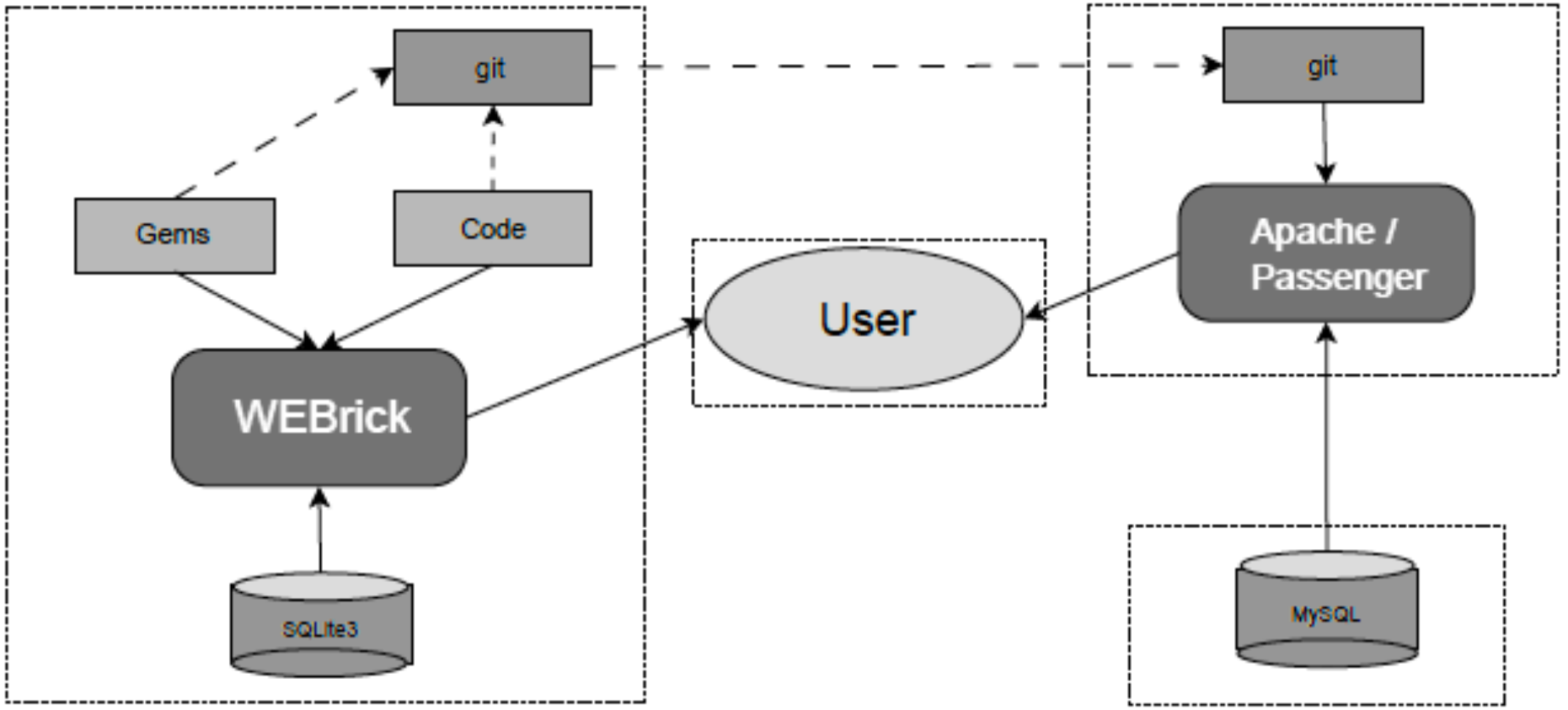

Figure 53—Application deployment road map

Deployment

- Automate deployment
  - It's a lot of work the first time around
    - One time tasks: building machines, installing and configuring software
    - Getting code into production
  - Make sure it doesn't stay that way
- Make it easy
- Make it repeatable
- Make it reliable

Automating Rails Deployment

- Set up production environment so it can check out project from version control system
  - Hopefully, this is already done for initial deployment
- Capistrano
  - Deployment management tool that works from the development environment
  - Needs remote access to production environment (i.e. via SSH key)
  - Issues remote commands to checkout and deploy new versions of project in production
    - Newest release lives in `current` directory
    - Alerts server to change by touching the `tmp/restart.txt` file

Every Production Environment is Different

- Different types/numbers of machines with varying CPUs, memory, disk space
  - Try to standardize for your particular project
- Different concerns for different applications
  - Speed
  - Security
  - Reliability
- Example: Christianbook.com environment

Monitoring a Production Application

- Internal monitoring approaches
  - Log analysis
  - Monitoring software (i.e. Nagios)
  - Security testing packages
- External monitoring
  - Web analytics packages (i.e. Google Analytics)
  - Third party testing services
    - Synthetic data vs. real user metrics (RUM)
  - Security testing services
- Monitoring at Christianbook.com

Web Performance

- “Premature optimization is the root of all evil…” Knuth
- Once the system is running and stable, we might need to optimize it
- Areas to tune
  - Back end – optimize the hardware and software serving the website
  - Front end – optimize the user’s experience by taking advantage of browser tools and techniques
    - Optimization can be actual or perceived

Back End Tuning

- Database – connection pooling, query optimization, hardware upgrades
- Web server – increase throughput
  - Enable more server processes/threads
  - Make requests process faster
  - Configuration tweaks and/or hardware upgrades
- Load balancing
- Caching
- Failover systems

Front End Optimization

- Browser caching – fewer requests for static resources
  - Set expiration on static resources that do not change often
  - Drawback: client must explicitly purge cache or wait for expiration to retrieve new versions of cached resources
- Compression of requests and responses
  - Browser and server deflate/inflate data to reduce bandwidth utilization
- Content delivery network (CDN) – distributed cache with nodes geographically "close" to end users
- http://webpagetest.org
- Google Page Speed

Web Analytics

- Need to measure how well website is doing
  - What are visitors doing on the site? Who are they? Are they converting? Why do they leave?
  - Why is the site/page slow/down?
- Web analytics systems and professionals investigate who visits a site and how they use it.
  - Google Analytics, Coremetrics, Omniture
- Instrumentation is the process of collecting (lower level) data on website processes and systems for troubleshooting and optimization purposes
  - statsd + Graphite

Milestone X