Computer Science 75 Lecture 0: January 26, 2009
Spring 2009 Andrew Sellergren
Scribe Notes

Contents

Computer Science 75
Spring 2009
Scribe Notes

Lecture 0: January 26, 2009
Andrew Sellergren

1 Welcome (0:00–5:00)

- O hai! Welcome to Computer Science E-75: Building Dynamic, Scalable Websites!
- This is a course in which students with *some* programming background will flourish and students with *no* programming background will also thrive if they are willing to work hard and bootstrap themselves.
- The course is structured around 5 projects, 4 of which we specify and 1 of which you come up with on your own and present at the end-of-the-semester Computer Science Fair.
- What do we mean by dynamic? Well, for starters, the websites we design will serve up more than just static content using HTML and CSS. The content of our websites will actually change based on the user who's viewing it.
- What do we mean by scalable? Building a website that supports a single user is a much easier (and less interesting) task than building a website that supports hundreds, thousands, or even hundreds of thousands of users. In this course, we'll be aiming for the latter. Throughout the semester, we'll be talking about the software and hardware required for this kind of large-scale infrastructure.

2 DNS (5:00–55:00)

- What actually happens when you pull up a web browser, type google.com into the address bar, and hit Enter? The host name, in this case google.com, is translated or *resolved* into a numeric IP address which is useful to computers.
- A useful analogy for Domain Name System (DNS) is the idea of specifying telephone numbers with memorable names. For example, 1-800-COLLECT is easier to remember than 1-800-265-5328 though they actually are one and the same. In the same way, example.com is easier to remember than 208.77.188.166, though they are actually one and the same. (Try clicking on both links to see for yourself!)
- As part of Project 0, you will purchase your own domain name so that your websites for this course will have a much cooler address than fas.harvard.edu/ username. And don't worry, we'll take care of hosting your websites on the course's server!
- GoDaddy.com is the registrar we're recommending, but you're welcome to use whoever you want. Know that GoDaddy will try to upsell you at every turn, so you'll get a lot of good practice at clicking the ”No Thanks” link.

- Note that you can choose from a number of Top-Level Domains (TLD) when picking your hostname. This is the familiar last part of the web address (e.g. .com, .net, .org, .info). Some TLDs are restricted (e.g. .edu, .gov), but for the most part you have free choice if the domain name isn't already taken.

- Why don't most American websites have .us as their TLD? Well, simply because we took charge of .com![1] Websites around the world tend to have country-code top-level domains (ccTLD) like .aq for Antarctica or .va for Vatican City. Interestingly, .tv is actually a ccTLD for Tuvalu but is often used for vanity's sake to stand for television. (Check out cs75.tv!)

- Once you've bought a domain name, you're going to tell your registrar who your nameservers are. Ultimately, it will be the course's nameservers, i.e. ns1.cs75.net and ns2.cs75.net, that know all about your domain name and what IP address it maps to. However, you'll need to tell the world that it should ask our nameservers for this information.

- What if you didn't have a course affiliation? You can tell GoDaddy not what the nameservers are, but actually what the IP address is of your website.

- If all of your websites are being hosted on our course server, then all requests will contain the same IP address. How will it know which website to spit out? Thanks to virtual hosting, the request will contain not only an IP address, but also a single line of text in the HTTP header that represents what the user actually typed into the address bar. It might say, for example, Host: foo.com. We can see this in the HTTP header, actually, by using a Firefox extension called Live HTTP Headers. You can think of HTTP as the language spoken by web browsers and web servers at the application layer of TCP/IP.

- If we look at the HTTP headers which are generated when we type google.com into the address bar, we see that our request didn't bring back a website right away, but actually spawned another request (among many others) indicating that google.com is actually www.google.com. This might be a low-level technical decision on Google's part, but is more likely for branding reasons. That is, Google wants to be known as www.google.com, not google.com.

- Who is it that regulates what entities can speak directly to the root servers and program them with additional domain names? That is, who gives GoDaddy the right to assign domain names? The Internet Assigned Numbers Authority (IANA) is one such governing body that oversees IP address allocation.

[1] America is the best. Just ask this guy.

- After the course is over, if you want your website to live on, you can pay for web hosting yourself. One popular web hosting service is Dreamhost.

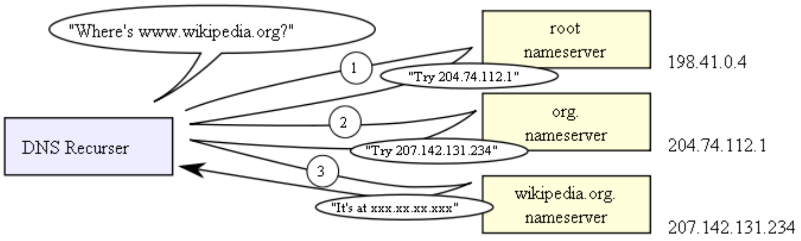
- One of the advantages of doing our own web hosting is that we have root access over the server. We've set up the server as a Virtual Private Server (VPS), which is to say that a single machine is partitioned so as to function as multiple servers. The resources of the machine, i.e. hard disk space and CPU cores and RAM, are likewise divided among multiple installations of the operating system, in our case CentOS 5. One neat trick which this enables is the ability to transfer an entire server from one machine to another without any downtime whatsoever.

- On top of the VPS, we run what's called a panel. cPanel is perhaps the most popular (despite being somewhat poorly designed) of these web-hosting control panels, but we use DirectAdmin, provided free for the university, which is just as reliable and much easier to use. The control panel will give you the ability to administrate DNS records so that each of your projects will live in a subdomain of your main website. That way, you can think of them as all having the same home directory. DirectAdmin provides a convenient GUI wrapper so that we can abstract away a lot of the messy details of Linux configuration files.

- There are two types of DNS records that you need to be concerned with in this course: an A record and a CNAME record. An A record, or address record, literally maps a domain name to an IP address. A CNAME record, or canonical name record, acts as an alias. The new domain name that you provide will be looked up after the first lookup. One downside of this is multiple DNS lookups, which can become hairy if your website experiences a lot of new traffic all at once. The upside, of course, is that the sysadmin doesn't have to change multiple DNS records (as he would if they were all A records) if the hardcoded IP address of the website changes.

- What's the deal with the supernews CNAME record in David's panel? We can see that it maps to cnn.com. Luckily, CNN doesn't filter out this type of thing, but it actually allows David to "reinvent" cnn.com, so to speak. If you navigate to supernews.malanrouge.com, notice that it simply displays the same home page as cnn.com. If we examine the HTTP headers, we can see that CNN spit out a HTTP 200 response, meaning OK. If they were concerned with branding issues, they could have spit out an HTTP 301 response (as Google did before with google.com), meaning Moved Permanently. That way, your web browser's address bar would display cnn.com instead of supernews.malanrouge.com.

- If you've ever been to a website that simply doesn't work if you don't add the www before its address, you should know that it's simply because the sysadmin didn't add one more DNS record (and one more line of code to the web server) that would map both to the proper IP address. Of course, nowadays, Firefox and some other browsers will do this lookup automatically.



- As the above diagrams conveys, DNS lookups are recursive. When you type in wikipedia.org into your address bar, your browser will begin by querying one of the 13 root nameservers[2] That root nameserver will spit out the IP address of the .org nameserver, which will spit out the IP address of the wikipedia.org nameserver, and so on until your browser finds the correct IP address.

- This diagram would suggest that the internet doesn't scale very well if there are only 13 root nameservers in the entire world. However, thanks to caching, these nameservers don't get queried all that often. Once wikipedia.org's IP address has been looked up once, that information will be stored by users or DNS servers or both so that in the future, time won't have to be wasted looking it up again. Luckily, this information doesn't change all that often. However, one downside is that if you as a sysadmin make a mistake and hardcode the wrong IP address into your DNS records, for example, this incorrect information might be cached for upwards of 24 hours.

- One other type of DNS record that is of interest to us is the MX record, or mail exchange record. As the name implies, this enables us to handle incoming mail on our servers. The value of 10 which is given here is actually a priority so that if you wanted to establish a hierarchy of web servers, you would need only create another DNS record with a different priority value.

- If you send an e-mail to something like john@harvard.edu, your e-mail client is going to go through the same series of DNS lookups that your web browser would. If you've ever gotten an error message saying that an e-mail was undeliverable but would be tried again in a few hours, it might have been because your client tried all the MX records and was then sleeping for a certain period of time before trying again. Mail is pretty resilient in this way thanks in part to DNS.

[2] That is, if this is the first time you've ever visited wikipedia.org. Come on, be honest, I know it isn't. It's probably not even the first time you've visited during this *class*.

- If we SSH (Secure Shell) to cs75.net using a client like PuTTY and open up a file called `/var/named/malanrouge.com.db` in a text editor such as Emacs or Vim, we can see many of the details we were viewing a few minutes ago using DirectAdmin. This is the config file for BIND (Berkeley Internet Name Domain). The entry for malanrouge.com looks like the following:

  ```
  malanrouge.com. 14400 IN A 64.131.79.130
  ```

  Notice the A which designates it as an A record as well as 14400, which is the Time To Live (TTL) in seconds. As you can see, the record looks very much the same as it did earlier on DirectAdmin, but thanks to the GUI wrapper, we don't have to worry about making simple syntax errors.

- A side note about SSH and SFTP: you'll be using these protocols to communicate with cs75.net. They will enable you to develop directly on the server and transfer files that you've developed elsewhere onto the web server when you want them to go live. We encourage you to develop on your home machine using XAMPP, a software bundle that includes Apache, MySQL, PHP and Perl. With this bundle, you can use your own computer as a web server and test out your web pages before you make them live and public by putting them on the course's web server.

3 More About the Course (55:00–78:00)

- In terms of prerequisites for the course, we assume only that you have *some* prior programming experience in a high-level language such as Java, JavaScript, PHP, Perl, C, C++, C#. If you only have experience making websites using HTML and CSS or programs like Dreamweaver, then you'll have to work a little harder, but you can still ultimately succeed in this course.

- All lectures will be videotaped and available online, usually within 72 hours of being so eloquently delivered by David in person.[3]

- The expectations of the course are to watch all lectures and to complete 5 projects, 4 of which we will assign and 1 of which you will design yourself. Project 0 is very simple and only requires you to setup your domain name and make a very basic website. Projects 1, 2, 3, and 4 will be the more time-consuming ones.

- What will you take away from this course? Today we've covered the basics of HTTP and next time we'll introduce you to PHP which will empower you to create dynamic websites. Soon after, we'll hit upon XML. For Project 1, this will enable you to implement a flat-text database file that can then be parsed by your website in order to display the menu of Three

[3] (scoff).

Aces Pizza. After XML (and a brief discussion of XPath and the DOM), we'll get into the more sophisticated database language called SQL. We'll follow that up with in-depth discussion of JavaScript and AJAX. Finally, we'll get into security issues.

- Sections! We'll hold one or more discussion sections each week, at least one of which will be held on campus and at least one of which will be recorded and posted online for distance students to benefit from. This is an opportunity to get more intimate exposure to the material (and the staff!).

- Projects! Project 0, as mentioned, will simply get you set up on the course's VPS and teh interwebs. Project 1 will be your implementation of the Three Aces menu (or at least part of it) as well as an online ordering system. Project 2 will be your E*Trade-like website that allows users to register and buy/sell stocks using live CSV (Comma-separated Values) data from Yahoo Finance. Project 3 will be a mashup whereby you will combine Google Maps' API (Application Programming Interface) and Google News to display markers with news headlines centered around a location provided by the user. Lastly, the Final Project will be of your own design (after staff approval). On Monday, May 18, you will have the opportunity to display your Final Project at the Computer Science Fair. Mark your calendars!

- Books! Not required! Check out the Resources section of the course website for free supplemental materials. If you *do* believe that you would benefit from additional reading material, check out the Books section of the syllabus.

- Office Hours! Mostly by appointment, but even if you are remote, you'll be able to interact with us via the Virtual Terminal Room. This uses Elluminate's software, which will allow us to chat with you but also to screen share with you and even take control of your terminal, if necessary, to help with debugging and the like.

- The course website! Let it be your lodestar this semester!

- Although we the course staff will abstain from the religious browser wars,[4] we will at least advocate the use of Firefox for development purposes based on its array of free plugins which allow you to view HTTP headers traffic as well as a webpage's CSS in a convenient, readable fashion.

- If you have questions which don't involve your own work specifically (i.e. What's wrong with my code?), you should post them to the course's bulletin board. If you have a more private question, feel free to e-mail us at help@cs75.net.

[4] Firefox rules!

4 More HTTP, XHTML, and CSS (78:00–90:00)

- For security and scalability reasons, you might be concerned with blocking the transmission of certain headers. If, for example, some enterprising lowlife discovers that you're using a certain version of Apache in which a buffer overflow bug has recently been revealed, he might be able to exploit this. So why broadcast this knowledge in the first place? In a similar vein, you might want to prevent the "X-Powered by: PHP" header from being transmitted. Even if this header is only 20 bytes but your server is getting a million hits a day, that adds up to 20 MB of data that is needlessly being passed back and forth from your server.

- XHTML is the stricter version of HTML. In HTML, for example, you don't have to close tags or put quotation marks around attributes.

- This course won't focus on the aesthetics of websites so much as the underlying logic. Thus, we will teach you the basics of CSS, but we won't mark you down if your websites are simply hideous![5]

- We will expect that your XHTML for all projects is *valid.* To check this for yourself, you can use the W3C's validator.

- A note about cross-browser compatibility: we will require that your websites render properly on at least two of the major browsers. Thankfully, there's help for you out there! Check out the Yahoo User Interface (YUI) library.

- The only actual XHTML that we'll introduce today is that which implements forms:

  - Text Fields

    ```
    <input name="email" type="text" />
    ```

  - Password Fields

    ```
    <input name="password" type="password" />
    ```

  - Hidden Fields

    ```
    <input name="id" value="123" />
    ```

  - Checkboxes

    ```
    <input checked="checked" name=remember" type="checkbox" />
    ```

  - Radio Buttons

    ```
    <input name="gender" type="radio" value="F" />
    <input name="gender" type="radio" value="M" />
    ```

  - Drop-Down Menus

[5] Check out one of David's first websites.

```
<select name=state">
  <option value=""></option>
  <option value=MA"></option>
  <option value=NY"></option>
</select>
```

Forms will be the basis for almost all user input on your websites. Take a look at the code above and see if you can make sense of it on your own before we dive into it more fully in the coming weeks!