Project 0: PizzaML*

due by Wed 7/17, noon ET

Goals.

- Design your own data model.
- Introduce you to PHP, XML, and XPath.

wickedlocal.com

* Inspired by http://www.xml.com/pub/a/2001/02/28/rddl.html.

Academic Honesty

All work that you do toward fulfillment of this course's expectations must be your own unless collaboration is explicitly allowed (*e.g.*, by some problem set or the final project). Viewing or copying another individual's work (even if left by a printer, stored in an executable directory, or accidentally shared in the course's virtual classroom) or lifting material from a book, magazine, website, or other source—even in part—and presenting it as your own constitutes academic dishonesty, as does showing or giving your work, even in part, to another student.

Similarly is dual submission academic dishonesty: you may not submit the same or similar work to this course that you have submitted or will submit to another. Nor may you provide or make available your or other students' solutions to Project 0, Project 1, or Project 2 to individuals who take or may take this course (or CSCI E-75) in the future.

You are welcome to discuss the course's material with others in order to better understand it. You may even discuss problem sets with classmates, but you may not share code. You may also turn to the Web for instruction beyond the course's lectures and sections, for references, and for solutions to technical difficulties, but not for outright solutions to problems on projects. However, failure to cite (as with comments) the origin of any code or technique that you do discover outside of the course's lectures and sections (even while respecting these constraints) and then integrate into your own work may be considered academic dishonesty.

If in doubt as to the appropriateness of some discussion or action, contact the staff.

All forms of academic dishonesty are dealt with harshly. If the course refers some matter to the Administrative Board and the outcome for some student is disciplinary action, the course reserves the right to impose local sanctions on top of that outcome for that student that may include, but not be limited to, a failing grade for work submitted or for the course itself.

Grades.

Your code (CSS, HTML, PHP, XML, *etc*.) will be evaluated along the following axes.

Scope. To what extent does your code implement the features required by our specification?
Correctness. To what extent is your code consistent with our specifications and free of bugs?
Design. To what extent is your code written well (*i.e.*, clearly, efficiently, elegantly, and/or logically)?
Style. To what extent is your code readable (*i.e.*, commented and indented with variables aptly named)?

Discuss.

☐ Surf on over to `cs75.net/discuss`, logging in if prompted, and poke around if you haven't already! Anytime you have a question this semester (that's not, say, personal in nature), do search Discuss to see if your question has already been asked by some fellow student and, better yet, answered! If not, post away!

Do be mindful of the syllabus's policies on academic honesty. Posting snippets of code is probably fine; posting an entire file is probably not. If ever in doubt, it's probably best to keep your post "private to staff" by checking the appropriate checkbox. You are welcome to respond to fellow students' questions, but, again, do be mindful of the letter and spirit of the course's policies.

Recommended Reading.

☐ First dive into HTML5 at `http://diveintohtml5.info/`, a free online book written in, well, HTML5! You won't need to leverage all of HTML5's features for this or future projects, so feel free to skim or skip sections that aren't of interest.

☐ Next read up on PHP at `http://php.net/manual/en/langref.php`. And skim the style guides at `http://pear.php.net/manual/en/standards.php` and `http://framework.zend.com/manual/en/coding-standard.html` to familiarize yourself with some conventions.

☐ If unfamiliar with XML and/or parsing XML with PHP, read up on both at `http://www.ibm.com/developerworks/xml/library/x-xmlphp1/` and `http://www.ibm.com/developerworks/xml/library/x-xmlphp2/`, focusing particularly on PHP's SimpleXML API.

However, do **not** store any XML in a PHP file for this project, as IBM did for some (crazy) reason in Part 1's Listing 6 as well as others (using PHP's heredoc syntax).

☐ If unfamiliar with `git` (and specifically merges), learn how to version files via the seminar at `https://manual.cs50.net/Seminars#Git_Magic:_Versioning_Files_Like_a_Boss`. You might also find helpful the Git Community Book at `http://book.git-scm.com/`.

☐ Now it's time to download some software.

- ☐ If you don't have it already, install the latest version of Google Chrome from `http://www.google.com/chrome`, then install the Window Resizer extension from `https://chrome.google.com/webstore/detail/kkelicaakdanhinjdeammmilcgefonfh`.
- ☐ Install version 17a of the CS50 Appliance, per the instructions at `https://manual.cs50.net/CS50_Appliance_17a#How_to_Install_Appliance`. If you've not used the CS50 Appliance before, see `https://manual.cs50.net/CS50_Appliance_17a#How_to_Use_Appliance` to learn how to use it!

☐ Now that you have version 17a of the CS50 Appliance installed, start the appliance. Select **Menu > Internet > Google Chrome** inside the appliance and then visit your favorite website to confirm that the appliance has Internet access. (Suffice it to say your own computer must too!) Then open a terminal inside the appliance, per `https://manual.cs50.net/CS50_Appliance_17a#How_to_Open_a_Terminal`, or SSH from your own computer to the appliance, per SSH from your own computer to the appliance, per `https://manual.cs50.net/CS50_Appliance_17a#How_to_SSH_to_Appliance`, and update the appliance by executing the command below.

```
sudo yum -y reinstall appliance50
```

If the command fails, try it again after restarting the appliance (as via **Menu > Log Out > Restart**). If still no luck, turn to your partner or `cs75.net/discuss` for a hand!

☐ If unfamiliar with virtual hosts (vhosts), acquaint yourself at `http://en.wikipedia.org/wiki/Virtual_hosting`.

Then, in a terminal window inside the appliance, execute

```
sudo geany /etc/hosts
```

and add the following line at the bottom of the file that opens, then save and quit `geany`.[†]

```
127.0.0.1 project0
```

Recall that `sudo` executes a command as the super-user (*i.e.*, `root`), which is necessary in this case, since `/etc/hosts` is only writeable by `root`. The line you just added ensures that `project0` will "resolve" (as via DNS) to `127.0.0.1`, which is the appliance's "loopback address."

Next, confirm that you have a directory called `vhosts` in your (well, John Harvard's) home directory. Then create a `project0` directory within `vhosts`. Then create an `html` directory within `project0`. Then `chmod` all three, plus your home directory, `711`.

[†] No need to append `project0.localdomain` to that line.

Now create a file called `index.php` inside of `~/vhosts/project0/html/` containing the below:

```
<?= 'hello' ?>
```

Then chmod the file `600` (though it should be already) and visit `http://project0/` with Chrome inside the appliance. Be sure to type the `http://`, else you'll end up Googling "project0". You should be greeted with `hi`. If you instead see some error, best to retry these steps![‡]

If curious as to why all this works, take a peek at `/etc/httpd/conf.d/appliance50.conf`. That file maps all of the appliance's virtual hosts to `/home/jharvard/vhosts/` so that creating a new virtual host called, say, `foo` inside the appliance is as easy as creating `/home/jharvard/vhosts/foo/` and `/home/jharvard/vhosts/foo/html/` and editing `/etc/hosts` so that `foo` resolves to the `127.0.0.1`.[§] Thanks to virtual hosts, each of your projects can appear to live in the root of a webserver (as opposed to some subdirectory), just like a real site!

BitBucket.

☐ So that you have a place to store revisions of this and future projects privately, head to `https://bitbucket.org/plans` and sign up for a **FREE** account. Be sure to use a `.edu` address so that you're automatically upgraded to an unlimited student plan.[**]

☐ Log into your Bitbucket account and create a new, private repo as follows:

- ☐ Ensure that **Create new repository** is highlighted (in dark blue).
- ☐ Input a value of **project0** under **Name**.
- ☐ Ensure that **Private** is checked.
- ☐ Ensure that **Git** is selected under **Repository type**.
- ☐ Check both **Issue tracking** and **Wiki** under **Project management** if you'd like.
- ☐ Select **PHP** under **Language**.
- ☐ Input a value for **Description** and/or **Website** if you'd like.
- ☐ Click **Create repository**.

You should then find yourself at a page whose URL is `https://bitbucket.org/alice/project0`, where `alice` is your actual Bitbucket username. **Ignore the instructions about `clone`.**

[‡] If you instead see `goodbye`, you really did something wrong.

[§] You can also access the appliance's virtual hosts from a browser on your own computer if you'd like, but you'll first need to edit `/etc/hosts` (if running Mac OS or Linux) or `C:\Windows\system32\drivers\etc\hosts` (if running Windows) on your own computer similarly. Just be sure to replace `127.0.0.1` with the appliance's IP address, which is displayed at all times in the appliance's bottom-right corner. To edit these files with a text editor, you'll likely need to invoke `sudo` (if running Mac OS or Linux) or **Run as Adminstrator** (if running Windows).

[**] If you already have an existing account, you're welcome to use that for the course. You can apply to have it upgraded to an unlimited student plan at `http://www.atlassian.com/software/views/bitbucket-academic-license.jsp`.

☐ Open a terminal inside the appliance or SSH to it from your computer. Then execute the following command to generate an SSH key pair (one public key, one private key):

```
ssh-keygen
```

Hit Enter at each of the prompts that appears (unless you'd prefer to secure the private key with a keyphrase). If you then execute

```
ls ~/.ssh/
```

you should see that you now have (among others) files called `id_rsa` (your private key) and `id_rsa.pub` (your public key). Execute

```
geany ~/.ssh/id_rsa.pub
```

then highlight and copy the contents of that file (which happens to be your public key). Visit `https://bitbucket.org/account/` and paste the key into the text field below **SSH keys**, then click **Add key**.

☐ Now let's push an initial commit to your repository. Again in a terminal window within the appliance, execute these commands, where `Alice` is your own name and `alice@example.com` is your own email address:

```
cd ~/vhosts/project0/
git config --global user.name "Alice"
git config --global user.email "alice@example.com"
git init
git add --all
git commit -m "Initial commit"
```

Then execute the below to add a "remote" for your Bitbucket repo, where `alice` is your own Bitbucket username:

```
git remote add origin git@bitbucket.org:alice/project0.git
```

Now push your code to that remote:

```
git push -u origin master
```

If you visit `https://bitbucket.org/alice/project0/src`, where `alice` is your Bitbucket username, you should see the code you just pushed.

☐ Henceforth, anytime you make changes to code that you'd like track in version control, execute the below (or similar, if more experienced with `git`) inside of `~/vhosts/project0/`:††

```
git add --all
git commit -m "a description of the changes you've made"
git push
```

Delicious XML.

☐ Just the other day, while waiting in line for lunch at your favorite pizza place, you were going on and on (as you often do) with a friend about how you're taking some course on building dynamic websites. "Maybe you should make this place a website so that we don't have to stand in line anymore," your friend interrupted, with just a hint of sarcasm. "Then we could order online."

"Hmmm," you replied, missing the sarcasm. "That is a fantastic idea!"

And so was born your Project 0. Your mission for this project is to implement a website that allows customers to place orders online!

Included at this document's end is a menu from a local haunt called Three Aces (who, sadly, has gone out of business, but let's pretend they're still with us). Turns out they sell more than just pizza. In fact, they offer ten different "categories" of food: **Pizzas**, **Specialty Pizzas**, **Special Dinners**, **Side Orders**, **Salads**, **Spaghetti or Ziti**, **Home made Lasagna Ravioli or Manicotti**, **Homemade Calzones**, **Wraps**, and **Grinders**. Some items only came in one size, but others clearly come in both small and large sizes, at different prices no less.

Your first challenge is to come up with a data model for this menu. You thought about using a database, but that feels like overkill, since you'd then also need to implement an interface with which the folks at Three Aces could update the menu. After all, you don't want a phone call every time they want to raise prices! Plus, the goal here is to save time. An XML file, then, feels like the right choice for this menu; your site will simply read items from it. That way, too, the folks at Three Aces can pretty easily update their own menu themselves with any old text editor. Of course, they'll have to keep the XML well-formed, but that seems a reasonable price to pay for an otherwise free website!

Spend some time thinking about how best to represent this menu as XML, keeping these goals in mind:

- ☐ It must be easy for someone less technical than you to make changes to the menu. The XML should be straightforward to read and alter.
- ☐ You must somehow keep track of each item's category, name, price(s), size(s), and description, if any.

†† If you omit the `-m` flag when committing, `git` will prompt you for a commit message with `nano`. Once you've provided said message, you can save and quit `nano` with **ctl-x**, **y**, then **Enter**.

- ☐ You should avoid duplication of data. Just because Three Aces sells Tomato & Cheese pizzas in two sizes, that doesn't mean "Tomato & Cheese" needs to appear twice in your file!
- ☐ Your model should be extensible. If Three Aces eventually decides to sell medium pizzas, they shouldn't need to call you!

Before deciding on a model, though, best to read on so that you know how your XML will be used. The overall design and aesthetics of this site are ultimately up to you, but we require that your site meet some requirements.

Feature Requirements

- ☐ Your site must not display Three Aces's menu on one huge page but, rather, allow customers to browse the menu by category. It is fine to display multiple (but not all) categories per page. **Spaghetti or Ziti** and **Home made Lasagna Ravioli or Manicotti**, for instance, sound like they belong on the same page.
- ☐ You need not convert Three Aces's entire menu to XML, lest tedium take the fun out of design; three items per category suffice, so long as those triples make clear your overall design. However, we suspect you'll enjoy your site more if you input more than three items per category!
- ☐ Customers must be able to add items to a "shopping cart" whose contents persist until the customers check out or close their browsers. Customers must also be able to update quantities and remove items outright (without, *e.g.*, having to change some item's quantity to 0).
- ☐ When customers follow some link to check out, they must be informed of their order's total cost and thanked for their order.
- ☐ Your site should perform rigorous error-checking. Under no circumstances should we be able to crash your site or induce unreasonable behavior. Letting us input negative quantities so that Three Aces owes *us* money is not, shall we say, reasonable. We will bang on your code and try to find faults; do not let us succeed.

Technical Requirements

- ☐ You're welcome to develop your site on any computer using any IDE or text editor, even without using the CS50 Appliance, but you must ultimately ensure that it works within the CS50 Appliance at a URL of `http://project0/` when installed in `/home/jharvard/vhosts/project0/`.
- ☐ Only files that should be web-accessible should live in `project0/html/`; everything else should live in `project0/` or some (other) subdirectory therein.
- ☐ Your markup language should be valid (or "tentatively" valid) HTML5, as per `http://validator.w3.org/`, unless some feature of your site requires otherwise (for the sake of some browser); explain in HTML comments any intentional invalidities. Your HTML should also be as pretty-printed as possible. Your CSS need not be valid.
- ☐ Your PHP must be extensively commented and be as pretty-printed as possible.

- ☐ You may use a WYSIWYG editor to generate HTML and/or CSS that you would like to use in your site.
- ☐ If you integrate third-party CSS or JavaScript libraries into your project, cite their origin with comments.
- ☐ If you incorporate or adapt snippets of PHP code from the Web into your project (*e.g.*, examples from `php.net`), cite the code's origins with comments.
- ☐ If you incorporate images from the Web into your project, cite the images' with comments.
- ☐ Your website must appear and behave the same on the latest versions of at least two of these browsers:
  - ☐ Chrome
  - ☐ Firefox
  - ☐ Internet Explorer
  - ☐ Opera
  - ☐ Safari

So long as your site meets the foregoing requirements, you are welcome to interpret this specification as you see fit. Imagine, perhaps, an ideal site for Three Aces. Then go implement that. Or, at least, as much as you can!

And don't forget about `cs75.net/discuss`!

Exit Interview.

- ☐ Once done with your site, put together a readme in a text file called `README` that lives in `project0/`.

  Treat this readme as your opportunity not only to explain but to justify your design decisions. Tell us why you modeled your XML as you did. Tell us why you chose, say, select menus over radio buttons for some feature. Tell us with which two (or more) browsers we should evaluate your site. And give us an overall sense of how your site works (*e.g.*, tell us which files do what). But still be succinct; keep this readme to just a few paragraphs in length.

How to Submit.

- ☐ A few days prior to this project's deadline, instructions for submitting your work will be announced at `https://www.cs75.net/`. Be sure to look for those directions and then submit your work prior to this project's deadline.

Three Aces

1613 Massachusetts Ave
Cambridge, MA 02139
Btwn Mellen & Everett St

617 491-2884
617 491-2889

Pizzas

Sm/lg

Tomato & Cheese	5.50	9.75
Onions	6.85	10.85
Peppers	6.85	10.85
Broccoli	6.85	10.85
Fresh Garlic	6.85	10.85
Mushrooms	6.85	10.85
Fresh Spinach	6.85	10.85
Anchovies	6.85	10.85
Hamburg	6.85	10.85
Pepperoni	6.85	10.85
Sausage	6.85	10.85
Meatball	6.85	10.85
Bacon	6.85	10.85
Ham	6.85	10.85
Olives	6.85	10.85
Grilled Chicken	7.95	11.80
Hawaiian	7.95	11.80
2-way Combo	7.95	11.80
3-way Combo	8.90	12.80
Extra Cheese	1.25	1.85

Speciality Pizzas

$9.80 Sm / $15.80 Lg

Three Aces Special
Mediterranean *Sliced Tomatoes, Olives, Spinach, Fresh Garlic, Mozzarella & Feta Cheese*
Vegetarian *Sliced Tomatoes, Onion, Peppers, Mushrooms, Broccoli, Mozzarella*
Meat Lovers *Pepperoni, Hamburg, Sausage & Bacon*
Bbq Grilled Chicken *Choice Of Vegetables*
Grecian Supreme *Grilled Chicken, Feta Cheese, Tomatoes, Kalamata Olives*

Special Dinners

Chicken Wing Dinner	7.25
Gyro Plate	7.25
Chicken Finger Plate	7.25
3 Piece Chicken Dinner	7.25
Cheeseburger or Chicken Burger Plate	5.25
Double-cheeseburger Plate	5.75
Chicken Kabob Plate *With rice, salad & pita bread*	7.85
Steak Tips Dinner *Served with side salad & french fries*	8.25
Fish & Chips Dinner *Served with side salad, french fries & tartar sauce*	7.35

*All Dinners served with French Fries and Salad**

Side Orders

Onion Rings	2.60	2.95
French Fries	2.25	2.65
Spicy Fries	2.60	2.95
Chicken Wings		5.75
Buffalo Wings		5.75
Chicken Fingers		5.75
Mozzarella Sticks(7 Pieces)		4.50
Slice Cheese Pizza		1.65
Slice Pepperoni Pizza		1.85
Homemade Spinach Pie		3.10
Buffalo Fingers		5.75
Chicken Burger		2.75
Cheeseburger		2.75
Spanakopita		3.25

Salads

SM/LG

Garden	3.50	4.50
Greek	4.50	5.50
Antipasto	4.50	5.50
Chef	4.50	5.50
Tuna	4.50	5.50
Grilled Chicken	4.95	5.95
Kabob Salad *With grilled Chicken & feta cheese*	5.45	6.45

Spaghetti or Ziti

With Sauce	5.40
With Sausage	6.45
With Meat Ball	6.45
With Veal	6.45
With Chicken Cutlet	6.45
With Mushrooms	6.45
A La Three Aces *Sausage, Mushrooms, Bacon & Ham Topped with Sauce & Mozzarella Cheese*	7.25
Eggplant Spaghetti or Ziti Dinner *With Mozzarella Cheese*	7.25

Home made Lasagna Ravioli or Manicotti

With Sauce	6.25
With Sausage	7.25
With Meatball	7.25
With Veal	7.25
With Chicken Cutlet	7.25
With Mushrooms	7.25
Veggie Lasagna	7.25

All Pasta Dinners Served with Garlic Bread and Salad

Homemade Calzones

$7.35-lg

Vegetarian
Sausage
Ham & Cheese
Chicken Cutlet
Grilled Chicken
Meatball
Grecian
Veal
Eggplant
Steak
Italian *All the above come with mushrooms, peppers& onions*
Grecian *Fresh Tomatoes, Spinach, Feta*

Wraps $4.95

Turkey Club Wrap *Turkey, Cheese, Lettuce, Tomato, Onion, Mayo*
Chicken Cobb *Grilled Chicken, Bacon, Cheese, lettuce, Tomatop, Onion, Honey Mustard*
Greek Supreme *Feta Cheese, Black Olives, Lettuce, Tomato, Onion, Greek Dressing*
Crispy Chicken *Chicken Finger, Lettuce, Tomato, Onion, Honey Mustard*
Steak Wrap

Grinders

Sm/lg

Meatless	4.50	4.95
Hamburger	4.50	4.95
Cheeseburger	4.75	5.75
Meatball	4.75	5.75
Sausage	4.75	5.75
American	4.75	5.75
Veal Cutlet	4.75	5.75
Hot Pastrami	4.95	5.95
Italian	4.75	5.75
Genoa Salami	4.75	5.75
Ham	4.75	5.75
Tuna	4.75	5.75
Roast Beef	4.95	5.95
B.l.t. *Bacon, Lettuce, Tomato*	4.75	5.75
Sliced Turkey	4.75	5.75
Three Aces Special *Turkey, Roast Beef & Bacon*	5.30	6.40
Cheese Steak	5.10	6.00
Onion Steak	5.20	6.20
Pepper Steak	5.20	6.20
Mushroom Steak	5.20	6.20
Special Steak	5.40	6.40
Steak Bomb *Each Additional Item on Steak Subs $0.25/ $0.40*	5.50	6.50
Pepper & Egg	4.50	4.95
Ham & Egg	4.75	5.75
Steak & Egg	5.60	6.50
Bacon & Egg	5.30	6.40
Chicken Cutlet	4.75	5.75
Eggplant & Cheese	4.75	5.75
Gyro On Pita		5.50
Grilled Chicken On Pita		5.25
Grilled Chicken Delight	5.50	6.50
Grilled Chicken Sub	5.25	5.85
Chicken Finger Sub	4.60	5.60