Amadon

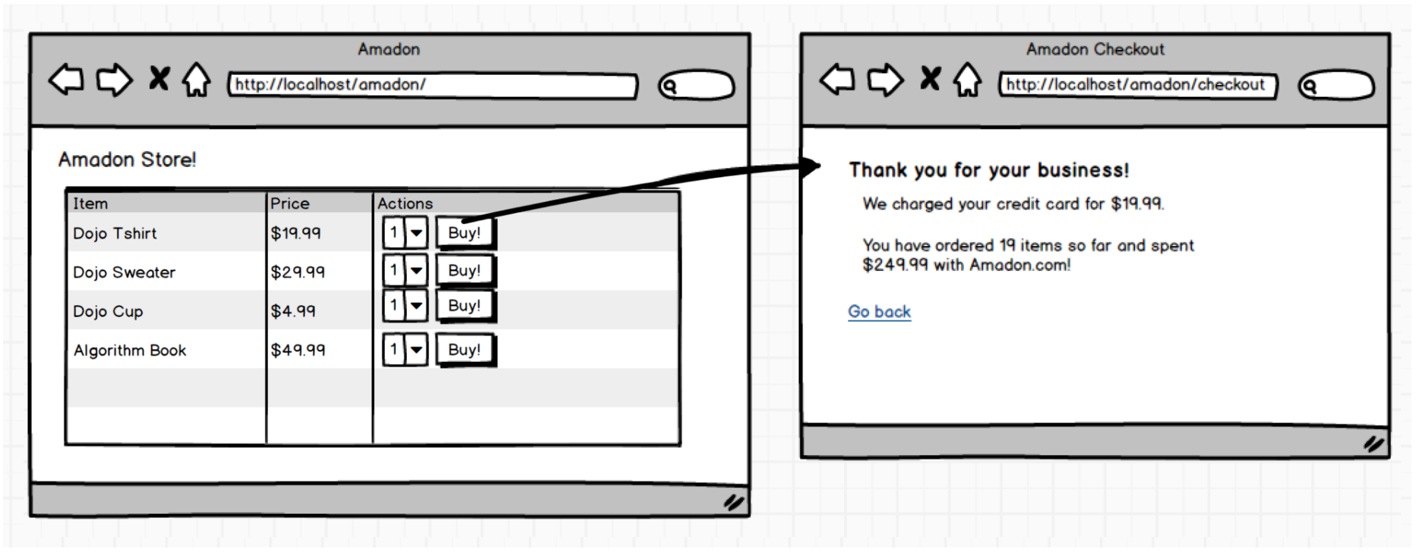
Objectives:

- Practice handling POST data
- Avoid rendering after a POST request
- Be careful about what you put inside <form>tags

We've decided to start building our own e-commerce site called Amadon.

The goal of this assignment is not to build a full-fledged e-commerce site (i.e. no login, validation, etc.). Rather, we want to point out some important things to consider when building forms:

1. What we should put in our forms versus what should be handled by the server in the backend



Amadon Store!

Item	Price	Actions
Dojo Tshirt	$19.99	1 Buy!
Dojo Sweater	$29.99	1 Buy!
Dojo Cup	$4.99	1 Buy!
Algorithm Book	$49.99	1 Buy!

Thank you for your business!

We charged your credit card for $19.99.

You have ordered 19 items so far and spent $249.99 with Amadon.com!

Go back

IMPORTANT LESSON 1

Say the customer reloads the checkout page after purchasing an item. How happy would your customer be if they were charged again and received double their original order? Probably not very happy!

A good developer should not have a method handle both the POST data *and* render HTML. This is a very common mistake made by developers--we should always double-check that we haven't made this mistake ourselves!

Instead have the http POST request sent to one route, have that route handle the POST data, and then *redirect* to a new GET route which displays the thank you html. This way, even when the user reloads the thank you page, it will not re-process the submitted order.

IMPORTANT LESSON 2

Another reason we designed this assignment like this is for you to see how easy it is to manipulate the form. For example, say that the form for ordering a Dojo T-shirt looked like this.

```
<form action='/amadon/buy' method='post'>
  {% csrf_token %}
  <select name='quantity'>
     <option>1</option>
     <option>2</option>
     <option>3</option>
  </select>
  <input type='hidden' name='price' value='19.99' />
  <button type='submit'>Buy!</button>
</form>
```

A somewhat sophisticated user could, for example, use the browser's Inspect Element tool to change the price to '0.01' and order lots of t-shirts for very cheap! A better way to handle this would be to have, for example, *product_id* as a hidden variable. This way, if they change the product_id using inspect element, they would just get a different item for their order.

In other words, have the form look more like below:

```
<form action='/amadon/buy' method='post'>
  {% csrf_token %}
  <select name='quantity'>
     <option>1</option>
     <option>2</option>
     <option>3</option>
  </select>
  <input type='hidden' name='product_id' value='1015' />
  <button type='submit'>Buy!</button>
</form>
```

Surprisingly, a lot of e-commerce sites are built where you could easily change the price. What if you built a web crawler/scraper to go through lots of e-commerce sites to specifically look for sites where price is part of the shopping cart form? You could reach out to them and tell them about the security flaw in their site. Maybe they'll hire you to make their site more secure? :)

Now it's your turn to go through this exercise and experience these issues for yourself. Follow the steps below to first experience these issues yourself, and then fix them. Follow the checklist below using this GitHub repository to get started.

https://github.com/TheCodingDojo/amadon

These same instructions are also provided in the GitHub repository's README file. In this code, we're not worrying about individual users, so we'll assume that all orders are being submitted by one user in order to calculate totals.

1. Clone the repository and peruse the code
2. Run makemigrations and migrate to create the necessary database tables
3. Seed the database with a few products (i.e. go into the shell and create 3-4 products)
4. Run the server and make a purchase
5. Add some basic styling (use Bootstrap or another CSS framework)
6. On the checkout page, calculate and display the total charge for the most recent order
7. On the checkout page, calculate and display the total quantity of all orders combined
8. On the checkout page, calculate and display the total amount charged for all orders combined
9. After making an order, hit the refresh button while on the checkout page and say yes/confirm. What do you notice?
10. Fix this issue so that users don't inadvertently make another order by mistake
11. Go back to the order form and use your browser's inspect element tool. Change the price of an item and then place an order. What do you notice?
12. Fix this issue so that users don't get to set the price of their items!