Tutorial 6
IERG 4210

Benedict Mak

Phase 3b Hint

1. Populate the main page's contents from DB with Handlebars or Express Handlebars
    - Populate the category list from DB
        - Read tutorial 5
        - Idea: Get JSON from pool.query, pass it to handlebars

Phase 3b Hint

- app.js

```
pool.query('SELECT “name”’, function (error, result) {
    res.render('home', { a: result.rows});
});
```

- home.handlebars

```
{{#each a}}{{name}}{{/each}}
```

Phase 3b Hint

1. Populate the main page's contents from DB with Handlebars or Express Handlebars
   - Based on the category picked by user, populate the corresponding product list from DB
     - Reflect catid in the URL (/?catid=1234)

```
app.get('/', function(req, res) {
  var catid = req.query.catid;
  res.send('from ' + catid);
});
```

Phase 3b Hint

- Reflect catid in the URL (/1234)

```
app.get('/:catid', function(req, res) {
  var catid = req.params.catid;
  res.send('from ' + catid);
});
```

- The corresponding product list is shown upon accessing the new URL in a new tab
    - Idea: Don’t make your website depends other anything but just GET parameters, and then use pool.query and pass it to handlebars

Phase 3b Hint

2. Populate the product page's contents from DB with Handlebars or Express Handlebars
    - Display the details of a product according to its DB record
    - Check slide 4

Phase 3b hint

3. Using JavaScript, dynamically update the shopping list
   - Users are allowed to update its quantity and delete it with a number input, or two buttons for increment and decrement
   - Idea: Number input - monitor for textbox change

```
$('input').bind('change keypress', function(e) {
    e.type //which event it trigger, keypress - every time after pressing key, change - after change and textbox is out focus
    update($this.val());
}
```

Phase 3b hint

- Idea: Buttons - monitor for click

```
$('btn).bind('click', function(e) {
    e.type //which event it trigger, click
    update(+1 or -1 or remove);
}
```

- Store its pid and quantity in the browser's localStorage

Phase 3b hint

```
var shopping_cart = { item: [ {'pid' : 1234, 'quantity': 1}, {'pid' : 2345, 'quantity': 1}] };

// Put the object into storage
localStorage.setItem('shopping_cart', JSON.stringify(shopping_cart));

// Retrieve the object from storage
var retrievedObject = localStorage.getItem('shopping_cart');

console.log('retrievedObject: ', JSON.parse(retrievedObject));

retrievedObject = JSON.parse(retrievedObject);

console.log(retrievedObject.item[1].pid);
```

- Get the name and price over AJAX (with pid as input)
- Calculate and display the total amount at the client-side

```
app.use('/pid', function (req, res) {
        var pid = req.query.pid;
        res.json({'name':'asdd', 'price':123 });
});
```

```
$.ajax({
        type: "GET",
        url: "./pid",
        data: { 'pid' : 123 },
        success : function(text)
        {
                alert(text.price*3);
        }
});
```

Phase 3b Hint

- When the addToCart button of a product is clicked, add it to the shopping list
  - Adding the same product twice adds up quantity, do not display two rows of record
    - Idea: Get pid from a hidden field

```
<input type="hidden" value="my hidden value" name="secret">
<input type="button" onclick="alert(this.parentNode.elements['secret'].value);"
    value="Add this product to your cart">
```

    - Idea(Cont'd) : Listen for button click, check localstorage for any rows with same pid, if no, add it to localstorage, and generate the HTML again

Phase 3b Hint

- Once the page is reloaded, the shopping list is restored
  - Page reloads when users browse another category or visit the product detail page
  - Populate and retrieve the stored products from the localStorage
    - Idea: check localStorage for the shopping cart record every time you reload the page

- Include a README.md file in your repo and document your application URL
  - remember to include the URL