

Lab 7 – Shopping-cart and cookie

Aim

The aim of this lab is to learn the shopping-cart and cookie.

Tips:

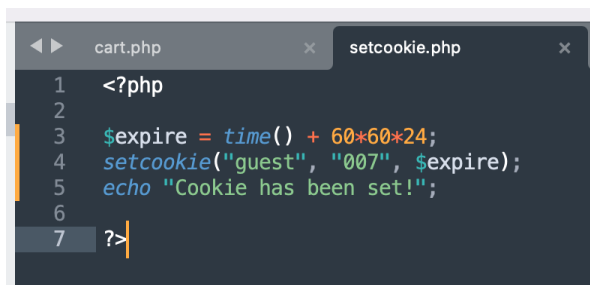
1. If you are not sure why you are doing something, ask a TA. This is what they are here for.
2. The M-Dev-Store online videos are good references while our labs have different focus. If you want to be an expert, you are recommended take both labs and on-line videos.
3. The forums @ LMO are available for questions and discussions.
4. These labs are expected take more than the 2 allocated hours. You should complete them in your own time before the next lab. Practice makes perfect!

Web Cookie:

1. Due to the security reason, the browsers work like a sandbox that isolate the webpage code from local computer resources. For example, webpage code cannot save the data into a local file. You may hear “browser cookie”. Cookies are small pieces of information websites store on your computer (s special kind of files). Your web browser stores and manages cookies. Websites are only allowed to look at their own cookies – for example, when you visit “www.xjtlu.edu.cn”, the code can’t examine cookies from other websites (i.e., www.google.com). Since HTTP is stateless, it is no way to do the continuously interactions without cookie.

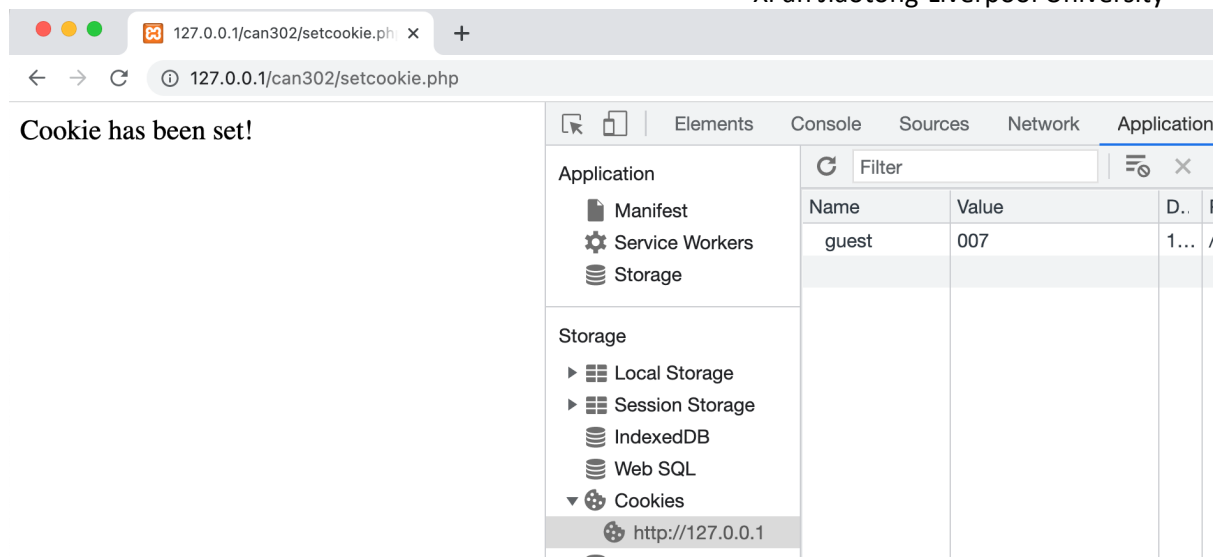
Usually each shopping-carts would belong to a specific customer. To identify the specific customer, everyone should have a unique credential at the e-store, for example – username or user_id. Till now, our demo does not have “users”. While some e-store would like users can freely browser products. Only till they decided to make an order, they need to register. Then the shopping-cart need to support both registered and anonymous users. We need to use cookie to mark the unique user.

The following code is an example for cookie:



```
1 <?php
2
3 $expire = time() + 60*60*24;
4 setcookie("guest", "007", $expire);
5 echo "Cookie has been set!";
6
7 ?>
```

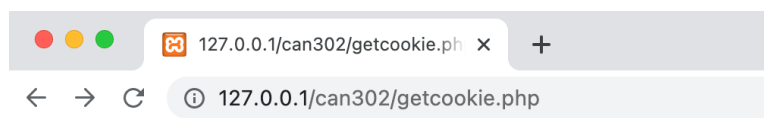
The cookie needs to have an expire time. Visit this page would generate a cookie and saved by the browser. You can easily find all cookies including the above one in your browser. It is as:



Then you can get it back before the expire time by the following code:

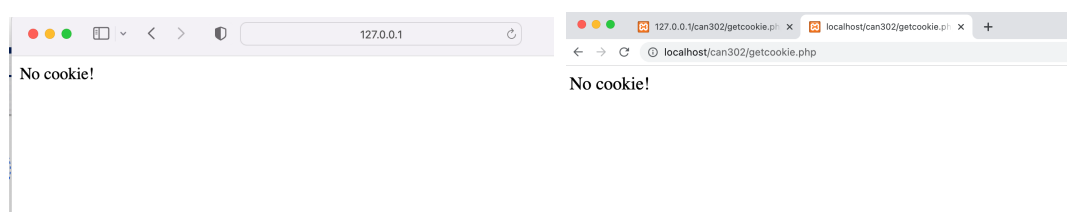


You will see the webpage as:



Fetch the guest name from cookie: 007!

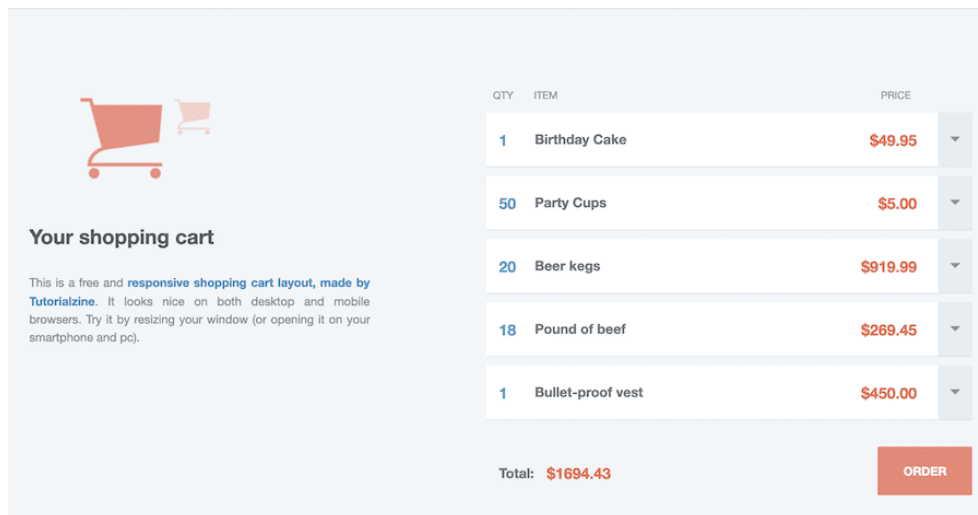
If you try to visit the `getcookie.php` with another browser or domain name, it will be as following:



You may try to figure out the reason by yourself.

Prepare the HTML code for shopping-cart:

- A typical shopping cart should contain the product, quantity, sub-total and total price. A free templet (<https://tutorialzine.com/2014/04/responsive-shopping-cart-layout-twitter-bootstrap-3/>) is as following:



We can directly integrate it to our lab. We just need to download the CSS file to our project and the static HTML code is as:

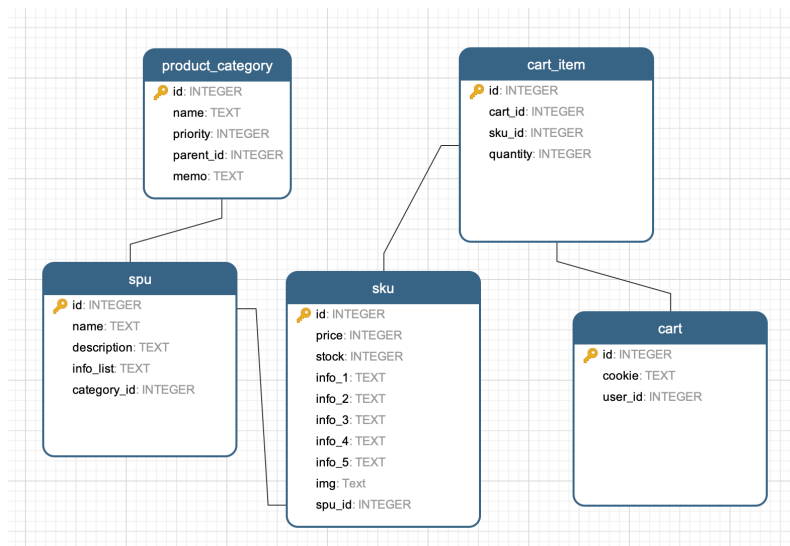
```

137 <div class="container-fluid text-center"> <!-- Show products here-->
138
139 <h2>Your shopping cart</h2>
140 <div class="line"></div>
141
142 <div class="col-md-5 col-sm-12">
143 <div class="bigcart"></div>
144 <p>
145   This is a free and <b><a href="https://tutorialzine.com/2014/04/responsive-shopping-cart-layout-twitter-bootstrap-3/"
146   title="Read the article!">responsive shopping cart layout</a></b>. It looks nice on both desktop
147   and mobile browsers. Try it by resizing your window (or opening it on your smartphone and pc).
148 </p>
149 </div>
150 <div class="col-md-7 col-sm-12 text-left">
151 <ul>
152 <li class="row list-inline columnCaptions">
153 <span>QTY</span>
154 <span>ITEM</span>
155 <span>Price</span>
156 </li>
157 <li class="row">
158 <span class="quantity">1</span>
159 <span class="itemName">Birthday Cake</span>
160 <span class="popbtn"><a class="arrow"></a></span>
161 <span class="price">$49.95</span>
162 </li>
163 <li class="row">
164 <span class="quantity">50</span>
165 <span class="itemName">Party Cups</span>
166 <span class="popbtn"><a class="arrow"></a></span>
167 <span class="price">$5.00</span>
168 </li>
169 <li class="row">
170 <span class="quantity">20</span>
171 <span class="itemName">Beer kegs</span>
172 <span class="popbtn"><a class="arrow"></a></span>
173 <span class="price">$919.99</span>
174 </li>
175 <li class="row totals">
176 <span class="itemName">Total:</span>
177 <span class="price">$1694.43</span>
178 <span class="order"><a class="text-center">ORDER</a></span>
179 </li>
180 </ul>
181 </div>

```

Data structure for shopping-cart:

3. Since we would like to support the guest user, we need a separate table for cart. It can be linked to both user (in the future) and cookie as we mentioned earlier. We need one more table goods to record the goods in a certain cart. The database now is as:



Do the shopping cart:

4. There are a couple of issues need to addressed:

- a) If it is a new guest or an old one?

Try to figure out the answer by the cookie. For a new guest, a new cart should be allocated.

A sample code is as:

```

40
47 function newcookie(){
48     $expire = time() + 60*60*24;
49     $random = md5($_SERVER['REMOTE_ADDR'].time());
50     setcookie("guest", $random, $expire, "/");
51     return $random;
52 }
53
54 // get user credential from cookie.
55 if(isset($_COOKIE["guest"])){
56     $guest = $_COOKIE["guest"];
57 } else {
58     $guest = newcookie();
59 }
60
61 // get the cart id of current guest.
62 $sql = "SELECT * FROM 'cart' WHERE cookie='{$_COOKIE["guest"]}';
63 $row=$con->query($sql)->fetch();
64 if(!$row){
65     $sql2 = "INSERT INTO 'cart' ('id', 'cookie', 'user_id') VALUES (NULL, '{$_COOKIE["guest"]}', 0)";
66     $con->exec($sql2);
67     $con->query($sql);
68     $row=$con->query($sql)->fetch();
69 }
70 $cart = $row["id"];

```

- b) If the guest adds new product to the shopping cart or only views the shopping cart?

The GET parameter with sku_id was used in last to add the selected sku into cart. It is a flag to determine the above difference. A sample code is as:

```

72 // add the sku to cart if it is not in the cart.
73 if(isset($_GET['sku'])) {
74     $sku = $_GET['sku'];
75     $sql = 'SELECT * FROM cart_item WHERE cart_id="'.$cart.'" AND sku_id='.$sku;
76     $item = $con->query($sql)->fetch();
77     if(!$item){
78         $sql2 = 'INSERT INTO `cart_item` (`id`, `cart_id`, `sku_id`, `quantity`) VALUES (NULL, "'.$cart.'", "'.$sku.'", 1)';
79         $con->exec($sql2);
80     }
81 }

```

Then, with the cart_id, we can fetch the product name and price and use the PHP code output them to the webpage. The sample code is as:

```

83 function myitem($db_con, $cart_id){
84     $sql = 'SELECT * FROM cart_item WHERE cart_id='.$cart_id;
85     $query = $db_con->query($sql);
86     foreach($query as $row){
87         $sql2 = 'SELECT * FROM sku INNER JOIN spu ON sku.spu_id=spu.id WHERE sku.id='.$row["sku_id"];
88         $query2 = $db_con->query($sql2)->fetch();
89         echo <<<EOT
90         <li class="row">
91             <span class="quantity">{$row["quantity"]}</span>
92             <span class="itemName">{$query2["name"]}</span>
93             <span class="popbtn"><a class="arrow"></a></span>
94             <span class="price">{$query2["price"]}</span>
95         </li>
96         EOT;
97     }
98 }
99

```

Home work

5. The current design would generate many DIRTY data for cart. Once the user aborts the shopping in middle, that cart is useless. One way to solve that issues is store all cart related info for guest in cookie. You can make a try.
6. The default quantity for each product is only 1. Try to use JS to adjust the quantity of each product.