


Dynamiczne zmienianie wartości pola z ilością zamawianych produktów i obliczanie ceny edytowanego zamówienia.

TWÓJ KOSZYK

#	Okładka	Tytuł	Ilość	Cena	Usuń
1		Krew na śniegu	<input type="text" value="1"/>	24.99 zł	×

Całkowity koszt zamówienia: 24.99 zł

Zapłać i zrealizuj zamówienie

Rysunek 1. Zrzut ekranu aplikacji przedstawiający koszyk sklepu internetowego.

Wykorzystane technologie: język PHP, JavaScript i biblioteka jQuery, format JSON.

```
$(document).on('blur', "td[contenteditable=true]", function()
{
    var message_status = $("#status_change_q");
    var field_userid = $(this).attr("id");
    var value = $(this).text();
```

Listing 1. Pierwszy fragment skryptu jQuery.

Kliknięcie na element z atrybutem **contenteditable** daje użytkownikowi możliwość jego edycji. Po wprowadzeniu nowej ilości produktów w zamówieniu następuje zdarzenie **blur**, które wykonuje się podczas kliknięcia poza wybrane pole lub wciśnięcie klawisza Tab.

Do zmiennej *message_status* przypisany jest element HTML o identyfikatorze *status_change_q*, który wyświetli powiadomienie o prawidłowej zmianie ilości produktów. Zmienna *field_userid* pobiera z edytowanej pozycji w koszyku jej identyfikator określony w postaci np. *quantity:1*, gdzie pierwsza wartość to nazwa tabeli w bazie danych którą będziemy aktualizować, a druga to identyfikator pojedynczego zamówienia. Do zmiennej *value* zostanie zapisana nowa wartość wprowadzona przez użytkownika.

```
$.ajax( {
    type: "POST",
    url: "edit_quantity.php",
    data: field_userid + "=" + value,
    success: function (msg)
    {
        message_status.show();
        message_status.html(msg);
        setTimeout(function () {
            message_status.hide();
        }, 300);
```

Listing 2. Drugi fragment skryptu jQuery.

Następnie wykonuje się asynchroniczne żądanie do serwera metodą POST, korzystając z funkcji **ajax**, która zaimplementowana jest w bibliotece jQuery. Zmiany z bazy danych zostaną obsłużone w pliku *edit_quantity.php*. Wysyłane dane mają postać `field_userid + "=" + value`, które posiadają wcześniej pobrane wartości. Jeżeli skrypt języka PHP zostanie wykonany, aplikacja wyświetli w odpowiednim polu informacje o pozytywnym lub negatywnym wyniku operacji i po chwili zniknie.

```
if (!empty($_POST)) {
    include "db.php";
    foreach ($_POST as $field_name => $val) {
        $val = (int) $val;
        if (is_int($val)) {
            $field_purchase = strip_tags(trim($field_name));
            $val = strip_tags(trim(mysql_real_escape_string($val)));

            $split_data = explode(':', $field_purchase);
            $purchase_id = $split_data[1];
            $field_name = $split_data[0];
```

Listing 3. Fragment skryptu PHP z pliku edit_quantity.php.

Na początku skryptu sprawdzamy czy przesłane do serwera dane posiadają nie są puste. Następnie zostanie wykonane połączenie z bazą danych. Sprawdzamy czy przesłana wartość z nową liczbą produktów jest liczbą całkowitą. Jeżeli tak, nazwa wysłanej zmiennej jest rozdzielana i odpowiednio do zmiennej *purchase_id* przypisujemy identyfikator zamówienia, a do *field_name* – nazwę tabeli którą będziemy aktualizować.

```
if (!empty($purchase_id) && !empty($field_name) && !empty($val)) {
    mysql_query("UPDATE `purchase_item` SET `$field_name`='$val' WHERE
`purchase_id`='$purchase_id'") or mysql_error();
```

Listing 4. Fragment skryptu PHP z pliku edit_quantity.php.

Następnie wykonywane jest zapytanie SQL, które aktualizuje rekord z ilością danej książki w danym zamówieniu.

Dalsza część skryptu napisanego w języku JavaScript, wykonuje drugie asynchroniczne żądanie do serwera, które zostaje zrealizowane jeżeli pierwsze zwróci wartość **success**.

```
$.ajax(
    {
        type: "POST",
        dataType: 'json',
        url: "current_price.php",
        success: function (data){
            $('#price_shopcard').html(data.price);
        }
    });
```

Listing 5. Trzeci fragment skryptu jQuery.

Drugie żądanie **Ajax** wykonuje skrypt zawarty w pliku *current_price.php*, którego zadaniem jest obliczenie aktualnej ceny całego zamówienia. Otrzymane dane z serwera przedstawione zostaną w formacie JSON. Po pomyślnym wykonaniu żądania, w miejsce elementu o identyfikatorze *price_shopcard* dynamicznie zostanie wstawiona nowa cena zamówienia z koszyka określonego klienta.

```
if ($_SESSION['auth'] == true && $_SESSION['login'] == true) {
    $price = 0;
    $user_id = $_SESSION['user_id'];
    include "db.php";
    $query_pag_data = ("SELECT k.price, p.quantity FROM book k, purchase_status s,
purchase b INNER JOIN purchase_item p ON b.purchase_id = p.purchase_id and
b.user_id='$user_id' where s.status_id=b.purchase_status_id and p.book_id=k.book_id");
    $result_pag_data = mysql_query($query_pag_data) or die('Database error' .
mysql_error());
    if (mysql_num_rows($result_pag_data) > 0) {
        while ($row = mysql_fetch_array($result_pag_data)) {
            $price = $price + $row['price'] * $row['quantity'];
            $msg['price'] = $price;
        }
        $msg['elements'] = mysql_num_rows($result_pag_data);
        echo json_encode($msg);
    } else {
        $msg['price'] = 0;
        echo json_encode($msg);
    }
}
```

Listing 6. Fragment skryptu PHP z pliku *current_price.php*.

Na początku skryptu sprawdzamy czy istnieją zmienne sesyjne, które określają zalogowanego użytkownika. Następnie zapytanie SQL zwraca wszystkie książki znajdujące się w jego koszyku i oblicza ich cenę, którą przypisuje do zmiennej *price*. Funkcją **json_encode** konwertujemy tablicę *msg*, zawierającą nowe dane do formatu JSON i zwracamy jako odpowiedź do skryptu JavaScript.