University of Malta
Faculty of ICT
Computer Information Systems

Dynamic Website Shopping Cart

Student: Lazar Lazarević
Lecturer: Dr. Christopher Porter
Study unit: Web Programming
Study unit code: CIS 1053 Year 2013/2014

This application is developed in ASP.NET, using Visual Studio 2013

1. **Scripts**: Only default ASP.NET scripts are used and inline script in shoppingcart.aspx for validation purposes.

   Domain is: http://GuitarStore.somee.com

   Hosted by Somee.com, free ASP.NET 4.5 hosting.

   Structure: At the top of the website can be found main navigation links, home (Guitar store), all instruments, shopping cart and about page.
   On the left hand side, instrument types are listed, where each link gives only instruments of that type in new page. In instruments page, instruments with picture, brand, model and price will be listed. Link "add to cart" will add selected item to cart and redirect customer to the shopping cart page. In shopping cart page, a user can edit the quantity of the products. Press on the Update button will update the items as well as press on the Enter on the keyboard. Typing 0 or less in quantity textbox will remove the product from the cart.
   Finally, Checkout button can be clicked once the user fill all textboxes with valid data. Once done, one email is sent to the administrator, and another one to the email entered in Email text box. After that, user will be redirected to the "thankyou" page with all items from shopping cart printed and customer details with thanks.

2. External libraries used: None

3. **XML**: For this assignment, I used two xml files. One for products, and another one for categories. In instruments.xml, I stored the data about: Guitar identification number, Instrument type: Eletric guitar, acoustic… Guitar brand, Model, description and price

   Example of Instruments XML:

   <instruments>

   <guitarType model="electricGuitar">

   <item id="1">

   <InstrumentType>1</InstrumentType>

   <brand>Kramer</brand>

   <model>Richie Sambora</model>

   <desc>

   Kramer Richie Sambora, 1987, 3 humbuckersseymorduncand,

   floyd rose, maple neck, mahagony body

   </desc>

   <price>1500.00</price>

   </item>

In InstrumentTypes.xml I stored data about the guitar types

Example of InstrumentTypes XML:

<root>

<InstrumentTypes>

<InstrumentType>

<InstrumentTypeID>1</InstrumentTypeID>

<InstrumentTypeName>Electric Guitars</InstrumentTypeName>

</InstrumentType>

4. **Validation**: In ShoppingCart.aspx, I wrote a javascript script with functionvalidateTextbox() to validate the user input for order details, that is Name, Email and home address. None field can be empty, and email has to be valid ex. example@example.example.

```
functionvalidateTextbox() {
var Name = document.getElementById("<%=Name.ClientID%>").value;
var Email = document.getElementById("<%=Email.ClientID%>").value;
var Address = document.getElementById("<%=Address.ClientID%>").value;

varemailPat = /^(\".*\"|[A-Za-z]\w*)@(\[\d{1,3}(\.\d{1,3}){3}]|[A-Za-z]\w*(\.[A-Za-
z]\w*)+)$/
varEmailmatchArray = Email.match(emailPat);

if (Name == "") {
alert('Enter First Name');
return false;
            }

if (EmailmatchArray == null || Email == "") {
alert("Your email address seems incorrect. Please try again.");
return false;
            }

if (Address == "") {
alert("Address field cannot be empty");
return false;
            }
        }
```

Known problems:

One of the problems that occurs only on live server is that sometimes it prints product categories two times on side menu. This never occurs in local host.
Some html generated by server which gives this error: WholeInsert4.js (failed) net::ERR_BLOCKED_BY_CLIENT

ASP.NET files:
In folder “Models” are .cs files for data.cs (to create new instances of data structures), GuitarStoreXmlReader.cs to parse the XML documents and store data in data structures, Instrument.cs to hold data about each instrument parsed from XML, same for InstrumentType but only for types of instruments, Order.cs holds data about order made in checkout page, and ShoppingCartItem.cs holds data about each item added to the shopping cart.

In logic folder is ShoppingCartActions.cs, there are all functions responsible for shopping cart logic, that is add to shopping cart, remove item, get total amount, update item, find item etc.

I avoided to add a lot of comments in .cs files as the code is self-explanatory.

In instrument.cs are actually two classes, Instrument and InstrumentCollection to store parsed data from XML.