Web Engeneering - Assignment

Tobias Ruck with parts by Maxim Sermin

January 31, 2014, DE

Contents

1 Design

As we didn't know how a jewellery shop should look like, we examinated some existing ones (thejewellershop.com, pandora.net, etc.) and got an impression how the shop should look like.

A fancy looking logo was build into the header of the website. That header always stayed the same throughout browsing. So we designed an according logo for "Jemline" and adopted the idea of a fixed header. In general, we put relatively much thought into the design of the header. The main concept behind the design and layout of the header was to keep it all very cleaned up in order to make navigation fast and intuitive. Hence, we decided to keep the number of tabs down by only putting those things into the

header, which we thought would be used most frequently. So, why is the "History" tab in there? Because we thought that it would make the shop appear way more trustful, if we reminded the user, that Jemline actually has a history. Led by how almost every user-interacting software does it, you find the search-bar in the upper right corner.

The main pages of the different shops were always used to present some products to the user as soon as they entered the website. This was often done with an animated slide show. We thought this was a good idea, because it immediately caught the user's attention and therefor helped to generate interest for the webshop. We incorporated the idea of the slide show (whose animation later had to be removed due to w3c-validation) and later added a list of some random products from the database underneath it. The use of a sort of dark-red color as the predominant color-scheme of the website, on the other hand was just the result of trying out some different RGB-combinations.

As the page is located in front of a dark background, by means of contrast we decided to go with a plain white background to put the actual information on. The individual elements are separated from each other by rather thin lines and boxes. This pays off regarding the overall concept of the website, but also has the downside that it may become hard to see on some monitors.

2 Implementation

The website is written using PHP as server language and XML as language for the data base. On client side, the used langs are HTML, CSS and JavaScript. The PHP implementation consists of a large amount of files of which are contact.php product_base.php terms.php cart.php copyright.php language.php categories.php main.php sales.php category.php history.php sales_base.php checkout.php index.php product.php search.php which are descibed below.

2.1 index.php

Everything a customer can do on the website is handled via index.php, that is index.php includes the other files mentioned (via include) when needed. This is done with a switch statement that handles in every case statement the include of the according php-file. The user can now browse the page using index.php?page=<php file without ending>but can not include non -existing files, as these are falling through the switch statement.

The page is divided with various div -tags, each named with a unique class attribute, with which the alignment of the page is handled in a seperate CSS file. This is linked in the head -tag of index.php. The tag with the class "container" includes the main tags visible on the page. It has a fixed size of 1024 pixels as it is assumed that semi modern video devices are used.

The background image of the body element is set to a jpg image, which, however, is only visible when the page is rendered in a window which size is greater than or equal of 1024 px, making it an extra feature for modern display devices.

The next element in the hierarchy is header, which contains: a png image rendering the logo of the company, JemLine; an unsorted list rendered as tabs linking to the pages

categories, sales and history, where the first letter of each tab is larger than the remaining letters; a png image rendering a shopping cart which links to cart.php; a text input field to search the shop for products containing the words entered.

The next element is a div with the class content. This is dependent on the $_GET['page'] variable and will yield different elements for different pages. These will be described later in this documentation.

The very last element in the hierarchy is footer where a centered list of links to Terms, Copyright, Contact and Language is rendered.

2.2 main.php

This php script gets included by index.php and can be invoked using jemline/index.php?page=main , jemline/ , jemline/? or jemline/?page=main. This just contains a slide show made with fotorama and jQuery.

2.3 categories.php

This php script gets included by index.php and can be invoked using jemline/index.php?page=categories or jemline/?page=categories.

The script reads the file db/categories.xml which contains an xml document with the categories of the online shop and echoes for each category a div with the class category.

2.4 category.php

This php script gets included by index.php and can be invoked using jemline/index.php?page=category&cate name>or jemline/?page=categories&category=<category name>.

The script reads the file db/categories.xml and db/products.xml. It then iterates over the categories to find the category supplied by $_GET['category']. If not found, nothing will be printed, else all products will be iterated and if it is in the specified category, it will be rendered.

2.5 product.php

This php script gets included by index.php and can be invoked using jemline/index.php?page=product&id=- id>or jemline/?page=product&id=<product name>.

It iterates over the db/products.xml file and renders the product with the product id supplied by $_GET['id']. The user can add an item to the cart by clicking Add to cart, which links to ?page=cart

2.6 cart.php

This script displays on the one hand the current items in the cart and can be used for managing them and is on the other hand called for adding a new item to the cart.

If $_POST['add-product-id'] is set, the according product will be added to the cart, that is, if the product id is set in $_SESSION['cart'] then it gets incremented by 1, else

it gets set to 1.Similary, the user can change the quantity of a product, that is the according product id in the session array gets set to the specified variable. The sum gets calculated and echoed in a side panel. The cart gets iterated and the products in it are rendered.

2.7 checkout.php

The user can enter his personal information, which gets validated using JavaScript and regular expressions. This date then is stored in a separate xml file in orders/ . It does not needed to be validated on client side, SQL injection is impossible, and invalid data can be inserted using a separate POST request. This however does not impose a threat to the website, since the data will only be seen by the admin in an escaped and secure way. The admin can access them using the folder orders/.

3 Usability and capabilities

Works flawlessly with Firefox, Chrome and IE (the 3 most common browsers at the current time). Not yet tested on mobile devices due to restriction to local host by XAMPP. Not yet tested on operating systems other than Windows 7

Feedback is provided through according headlines on each page where it makes sense (e.g. the users are not told, that they are on the product-specific page, if they just clicked on a product). If necessary, it is also possible to look at the address-bar of the browser, as the php statements as well are meant to reveal some information about the content of the current page. Throughout the ordering process you are likewise provided with feedback about your current position in the process.

4 Potential improvements

There are various ways to improve the web shop.

4.1 Use a SQL-database

In our website, we used an xml-file to store the few sample products of Jemline. This way it was simpler to exchange it with each other (in case changes were made) and we did not need to run a MySQL-server alongside the apache. In addition, the only query we would have needed, would be the select-query. This could easily be simulated using PHP. Of course, if the website would be intended for further commercial use, a proper SQL-database would definitely the way to go.

4.2 More dynamic

While JavaScript remains a controversial topic in web-development, you certainly can use to make a website noticeably more interactive in a user-friendly-way.

For example you could already start searching through the database while the user is still typing things into the search-bar and display results to them on the fly (like google-search does it). Another possibility is to display all kinds of additional information in the form of message-boxes, but only while the user is hovering over the corresponding element on the website.

4.3 More security

Currently, the only validation for user input lies at client side. Since invalid data does not break the whole system, as XML is used, no checks have to be done. When moving to a more enterprise level, however, one has to check the data fed to the server by means of SQL-injection or Cross-site-scripting.