Fachhochschule Köln
Cologne University of Applied Sciences

Fachhochschule Köln
Fakultät für Informatik und Ingenieurwissenschaften

Documentation

WPF Modern Web

Vorgelegt an der Fachhochschule Köln
Campus Gummersbach
im Studiengang
Medieninformatik

ausgearbeitet von:
Dominik Schilling
(Matrikelnummer: ???)
Laura-Maria Hermann
(Matrikelnummer: 11083968)
Dario
(Matrikelnummer: ???)

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Introduction

In this documentation are the elaborated content of the WPF modern Web application. Is documented, among others, the source code but also approaches or considerations which have been made in advance. The task was to create a gallery in the form of a website, which had to include the following:

- The gallery as such should be implemented using Javascript.
- Individual images should be zoomed by clicking and navigable by arrows.
- There should be a log-in available area.
- MySQL connection to a LogIn for the others but also to add new images
- JSON, JQuery and AJAX should be used.

2 Planning

The first task was to think exactly how the gallery will look like - not optically but functional. Because a log-in area was prescribed it seemed sensible to realize an administrator who got an extended functionality (which is initially irrelevant)

When considering existing galleries on the net you could roughly distinguish between two groups.

On one hand, web pages which serve as a real gallery and only have one author who may publish works. This would be only an admin, the author himself who upload photos, delete, ...

On the other hand, many galleries will be held as a kind of platform. These are in addition to the administrator other users who register very simple and thus can upload their photos.

As part of the WPF's it was decided to create the second variant.

3 Realization

3.1 MVC-paradigm

First is to say, that was held during the entire implementation of the MVC paradigm. This is explained in more detail below and explain its components.

What is MVC?

MVC (model, view, controller) is an architectural structure of software, which provides a strict division of tasks and separation. A distinction is made between the three components model, view and controller. These are explained in more detail below.

Request

Example:

The Request class extracts the URL into its component parts. user, for example, is one segment.

Controller

The controllers include the logic always belongs to a resource.

Example:

localhost / user then the controller is loaded, see load.php. The controllers are located in includes / controllers directory. If no other resource specify, for example localhost / user / blaa is called the index method of the controller. If the resource was blaa and indicate the method exists, it is called when it does not exist, the index is retrieved.

Views

The views are located in the includes / views directory. The views are responsible for the visual representation of data.

Example call:

A view can consist of several template, eg headers - content - footer.

Model

3.2 Database

One of the first tasks is creating a database in which all uploaded images are stored and can be retrieved. In the following, all the contents and information related to the database are examined in detail.

3.2.1 Create

First, to create a database. Each user can later use the graphical interface to create a gallery and thus create a set of tables in the database. Following you are able to find explanations of all realized tables and fields. Collation: utf8_general_ci

3.2.2 Database tables

Users This is to inform the users of the gallery. These are uniquely identified by an ID.

Usermeta Metadatas of the users

Images Data of the images

Imagemeta Metadatas of the images

Gallery Informations about the gallery in general

3.2.3 SQL-Query

The following SQL commands are called to create the tables used commands.

```
CREATE TABLE `gallery` (
  `ID` bigint(20) unsigned NOT NULL,
  `user_id` bigint(20) unsigned NOT NULL,
  `is_public` tinyint(1) NOT NULL DEFAULT '1',
  `gallery_title` text NOT NULL,
  `gallery_description` longtext NOT NULL,
  PRIMARY KEY (`ID`)
) DEFAULT CHARSET=utf8;

CREATE TABLE `imagemeta` (
  `ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `image_id` bigint(20) unsigned NOT NULL,
  `meta_key` varchar(255) NOT NULL,
  `meta_value` longtext NOT NULL,
  PRIMARY KEY (`ID`),
  KEY (`image_id`),
  KEY (`meta_key`)
) DEFAULT CHARSET=utf8;

CREATE TABLE `images` (
  `ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `user_id` bigint(20) unsigned NOT NULL,
  `uploaded_date` datetime NOT NULL,
  `image_title` text NOT NULL,
  `image_description` longtext,
  `image_caption` text NOT NULL,
  PRIMARY KEY (`ID`),
  KEY (`user_id`)
) DEFAULT CHARSET=utf8;

CREATE TABLE `image_relationships` (
  `image_id` bigint(20) unsigned NOT NULL,
  `gallery_id` bigint(20) unsigned NOT NULL,
```

```
  PRIMARY KEY (`image_id`, `gallery_id`),
  KEY (`gallery_id`)
) DEFAULT CHARSET=utf8;

CREATE TABLE `usermeta` (
  `ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `user_id` bigint(20) unsigned NOT NULL,
  `meta_key` varchar(255) NOT NULL,
  `meta_value` longtext NOT NULL,
  PRIMARY KEY (`ID`),
  KEY (`user_id`),
  KEY (`meta_key`)
) DEFAULT CHARSET=utf8;

CREATE TABLE `users` (
  `ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `user_login` varchar(60) NOT NULL,
  `user_pass` varchar(64) NOT NULL,
  `user_email` varchar(100) NOT NULL,
  `user_registered` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `user_status` int(10) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY (`user_login`),
  KEY (`user_email`)
) DEFAULT CHARSET=utf8;
```

3.3 resources

There follows an explanation of the necessary resources.

3.3.1 /register

At this point, the registration of new users is possible. This only needs to be done so far when create a gallery and want to post pictures. Registration information such as name, password and email address are required.

3.3.2 /login

The direct login area if you are already a registered user.

- Login

3.3.3 /logout

The direct logout area if you have previously logged in.

- Logout

3.3.4 /profile

His own profile. At this point the user gets to make any profile settings and changes. This would be for example loading up an avatar.

- Setting
- Avatar
- Mail
- Name
- ...

3.3.5 /user/{$id}

With the help of this resource you are able to get the profile of a specific user (which is uniquely identified by its ID). This is especially useful for viewing all of the galleries of the user.

- User profile
- Overview of public / private galleries

3.3.6 /user/{$id}/galleries/{$id}

If you want to directly get a concrete gallery of a user, you have to specify the ID of the gallery as a query parameter.

- Gallery of a single user

3.3.7 /search

Use the search function, you can get special content. These are selected with filters.

- Search
- Filters

3.3.8 /install

At the beginning of a session, the installation must be carried out.

- Installation routine

3.4 Code

In the following, the program code is explained and illustrated.

3.4.1 Controllers

How to recognize some control classes were implemented and used.

Almost all of these controller classes differ only in their title. The page title, which is visible in the browser connection is "set" at this point.

For example, the title of 'class-error controller' on "404 - Page does not exists!" placed. Another was described in controller.

3.4.2 Form-Validation

Especially in all forms, such as the Login or Register, it was important to make some measures for validation.

Especially in all forms, such as the Login or Register, it was important to take some measures for validation. When logging in to be examined, first, whether the username is available and secondly whether the associated password is correct.

When registering on the one hand to make it, that the username and email are not forgiven, and on the other hand, ensure that both passwords entered are the same.

These errors treatments are implemented in the class "class-user-manager" and shown by the functions „validate_new_user“, „set_current_user“ and „create_user through“.

First, a variable is created for each field of the form. If this is no longer empty after entering of the user, the value is what the user has entered fetched and processed using the POST method. In addition, it is converted by the htmlspecialchars function. This is particularly important in order to maintain security. In general this function transform, with use of the selected flags, single characters, so that they are not interpreted as false. In this context, it is particularly the quotation marks, which are converted to to prevent misunderstandings as <input value="Mein"Name» (the value would only be "Meinänd Name"would be considererd as invalid HTML-Object).

3.4.3 CSS-Framework Bootstrap

As a basis and simplify the work was resorted to the free bootstrap framework.

This simplified, especially at the beginning, the realization and getting huge. Bootstrap has the advantage to be platform independent, among others but also support in its latest

version of the "responsive design". The site got a dynamic structure, regardless of which device you use.

3.4.4 Functions

In the following there are declarations about the used functions writen in the functions.php. In addition it should be said here that only roughly functionality is explained. Further descriptions can be found in the source code.

At first there was a function for checking the compatibility of the php-Version. Its important to tell the user, if it is like this, that he has an older php-Version or not started the MySQLi extension.

Another functions checks if the installation was already complete. If this isn't the case it redirects to the installation page.

To guarantee the safety of the passwords was implemented a hash function. For this purpose, the existing hash function framework was used.

3.4.5 Javascript

It follows a short description of the javascript functions which was implemented.

Password strength function As an extra to the user a feature was implemented, which determines the strength of the password