Fachhochschule Köln

Fakultät für Informatik und Ingenieurwissenschaften

Documentation

WPF Modern Web

presented at Cologne University of Applied Science
Campus Gummersbach

elaborated by:
Dominik Schilling
(Matrikelnummer: 11081691)
Laura-Maria Hermann
(Matrikelnummer: 11083968)
Dario Vizzaccaro
(Matrikelnummer: 11085033)

Inhaltsverzeichnis

1 Introduction

This documentation contains the content elaborated within the scope of the WPF modern Web applications. It includes, among other sections, the source code and the approaches or considerations that have been made during the early stage of the project. The goal was to create a gallery in form of a website, which had to include the following features:

- The gallery as such should be implemented using HTML5, CSS3 and JavaScript.
- Individual images should be zoomed when selected and navigable with arrows.
- A log-in area should be available.
- MySQL implementation to allow log-in for users, but also to allow the upload of new images.
- JSON, jQuery and AJAX should be used.

2 Planning

The first task was to discuss how the gallery should look like - not optically but functionally. Because a log-in area was required, it seemed reasonable to create an administrator with extended rights.

When looking at existing galleries on the internet it is possible to roughly distinguish between two groups.

On the one hand, there are web pages which serve as a real galleries and only have one author who can publish content. In this case only the admin - the author - himself can eg. upload, delete, edit, etc. photos.

On the other hand, many galleries are used as a platform. These web pages give not only administrators, but also registered users the possibility to upload and manage their photos.

Due to the WPF we had to choose the second alternative.

3 Realization

3.1 MVC-paradigm

The idea of administrators and users beeing able to upload and manage photos, was pursued throughout the entire implementation of the MVC paradigm. In the following this will be further explained in more detail.

What is MVC?

MVC (Model-view-controller) is an software architecture pattern which provides a strict separation between reprasentation of information and user's interaction with it. A distinction is made between the three components model, view and controller. These are explained in more detail below.

Request

Example:

```
GET localhost/user/?foo=bar
```

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

Controller

The controllers include the logic that always belongs to a resource.

Example:

```
localhost / user
```

then the controller is loaded, see load.php. The controllers are located in ïncludes / controllers"directory. If no other resource is specified, for example "localhost / user / blaa"the index method of the controller is called and will start. If the resource was "blaaänd the indicated method exists, it is called. When it does not exist, the ïndexïs retrieved.

Views

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

Example call:

```
$view = new View( 'install/success' );
$view->set_page_title( 'Installation' );
$view->render(); // Gibt das Template aus
```

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

Model

3.2 Database

One of the first tasks was 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 of all, a database was created. Each user can later use the graphical interface to create a gallery and thus create a set of tables in the database. In the following explanations of all realized tables and fields can be found. 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.

Field	Typ
ID	bigint(20)
user_login	varchar(50)
user_pass	varchar(64)
user_email	varchar(100)
user_registered	datetime
user_status	int(10)

Usermeta Metadatas of the users

Field	Typ
ID	bigint(20)
user_id	bigint(20)
meta_key	varchar(255)
meta_value	longtext

Images Data of the images

Field	Typ
ID	bigint(20)
user_id	bigint(20)
uploaded_date	datetime
image_caption	longtext
image_title	text

Imagemeta Metadatas of the images

Field	Typ
ID	bigint(20)
image_id	bigint(20)
metakey	varchar(255)
meta_value	longtext

Gallery Informations about the gallery in general

Field	Typ
ID	bigint(20)
user_id	bigint(20)
public	boolean
gallery_title	text
gallery_description	longtext

ImageRelationships Relations between images and galleries

Field	Typ
ID	bigint(20)
gallery_id	bigint(20)

3.2.3 SQL-Query

The following SQL commands are called to create the tables.

```
CREATE TABLE `galleries` (
  `ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `user_id` bigint(20) unsigned NOT NULL DEFAULT 0,
  `is_public` tinyint(1) NOT NULL DEFAULT 1,
  `gallery_title` text NOT NULL,
  `gallery_slug` varchar(200) NOT NULL DEFAULT '',
  `gallery_description` longtext NOT NULL,
  `gallery_created` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY (`ID`),
  UNIQUE KEY user_id_gallery_slug(`user_id`, `gallery_slug`)
) DEFAULT CHARSET=utf8;

CREATE TABLE `imagemeta` (
  `ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `image_id` bigint(20) unsigned NOT NULL DEFAULT 0,
  `meta_key` varchar(255) NOT NULL DEFAULT '',
  `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 DEFAULT 0,
  `image_uploaded` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `image_filename` text NOT NULL,
  `image_title` text NOT NULL,
  `image_description` longtext NOT NULL,
  PRIMARY KEY (`ID`),
  KEY (`user_id`)
) DEFAULT CHARSET=utf8;

CREATE TABLE `image_relationships` (
  `image_id` bigint(20) unsigned NOT NULL DEFAULT 0,
  `gallery_id` bigint(20) unsigned NOT NULL DEFAULT 0,
  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 DEFAULT 0,
  `meta_key` varchar(255) NOT NULL DEFAULT '',
  `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 DEFAULT '',
  `user_pass` varchar(128) NOT NULL DEFAULT '',
  `user_email` varchar(200) NOT NULL DEFAULT '',
  `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 was possible. This only needs to be done if wanting to create a gallery and 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 as an user.

- Login

3.3.3 /logout

The direct logout area if you have been previously logged in.

- Logout

3.3.4 /profile

A profile might look as followed. At this point the user can make any changes to his profile settings, within the range of his permissions. This would be eg. 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 specific gallery of an user, you have to specify the ID of the gallery as a query parameter.

- Gallery of a single user

3.3.7 /search

Using 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

Here is how to recognize the control classes that 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!" was placed. Another one was described in the controller.

3.4.2 Form-Validation

It was of great importance that measures of validation were implemented in all forms, such as the Login or Register forms. When logging in it has to be verified, whether the username is available and whether the associated password is correct.

In order to registrate successfully, an availible username and an unique email address are required. Furthermore, it has to be ensured that both passwords entered are the same.

These error handlings 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 the field is no longer empty after the user's interaction, the value is what the user has entered. Then the entered value is fetched and processed using the POST method. In addition, it is converted by the htmlspecialchars function. This is particularly important in order to maintain an adequate security level. In general this function transforms single characters with the use of selected flags, so that they are not interpreted as false. In this context, it is particularly important for quotation marks, which are converted to prevent misunderstandings as <input value="Mein"Name» (the value would only be "Meinänd Name"would be considererd as an invalid HTML-Object).

3.4.3 CSS-Framework Bootstrap

As a basis and as a simplification, the work was accomplished with the free Bootstrap 3.0 framework.

This simplified, especially at the beginning, the realization and the rapid progress. Bootstrap has the advantage, among others, to be platform independent, but also supports in its latest version the "responsive design". The site has a dynamic structure, regardless of which device was used.

3.4.4 Functions

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

Before the installation process begins, a function checks the compatibility of the installed php-Version. If it is the case, it's important to warn the user that he has an older php-Version or that the MySQLi extension is not started.

Another function checks, whether or not the installation was already completed before. If this isn't the case it redirects to the installation page.

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

3.4.5 Javascript

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

The scripts found in the first part of the javascript were taken from the respective project websites, eg. the script for the fancyBox and for the Carousel. These were open source packages which were available free of charge.

The most important file is the javascript image-uploader.js. In this file the entire process of image uploading and thus the actual upload functionality is implemented. Furthermore, both Ajax and JSON elements are introduced, as requested in the assignment.

First, the images are (this is new in HTML5) selected via a multiple-file upload input and loaded locally using the File Reader to then create a thumbnail. After the upload an AJAX request, which runs on a XML HTTP request of level 2 begins.

Password strength function As an extra for the user a feature was implemented, which determines the strength of the entered password. We used the JavaScript library named zxcvbn by Dropbox therefor.

4 Server requirements and installation

4.1 Server requirements

Before starting the installation process, please make sure that the server meets the following minimum requirements:

- PHP version 5.4 or greater
- MySQL version 5.0 or greater with the MySQLi extension running
- Running GD extension
- Sessions support has to be active

Please note that these settings are not only recommended, but required. Failing to meet the above mentioned requirements will result in an early interruption of the installation process.

4.2 Installation

The installation process is fully automated, but to make sure everything goes smoothly we have to configure a few parameters first. To do so, you will need a web server, a FTP client, a text editor and a browser of your choice.

1. Download the latest available version of the script (GitHub is recommended).[**?**]
2. Unzip the whole content of the file to your preferred location.
3. Create a MySQL database on your web server and an user with the rights for accessing and modifying it.
4. Upload the previously unzipped files to your web server.
5. Rename the .htaccess-sample and config-sample.php, respectively to .htaccess and config.php. At this point please note: If you have placed the files in a subdirectory, you will have to edit the .htaccess accordingly. Eg. if the subdirectory is called "gallery", you should modify line no. 7 "RewriteBase /"to "RewriteBase /gallery". If you have placed the files into the root of your domain (eg. http://yourdomain.com/) this step is not necessary.
6. Open the file config.php and search for line no. 18 $app−>config−>url = 'http://localhost';. Replace "localhost"with your domain, including the subdirectories. For the above example it would be $app−>config−>url = 'http://yourdomain.com/gallery;. Also change the paramenter in line no. 24 $app−>config−>segment_offset = 0; to

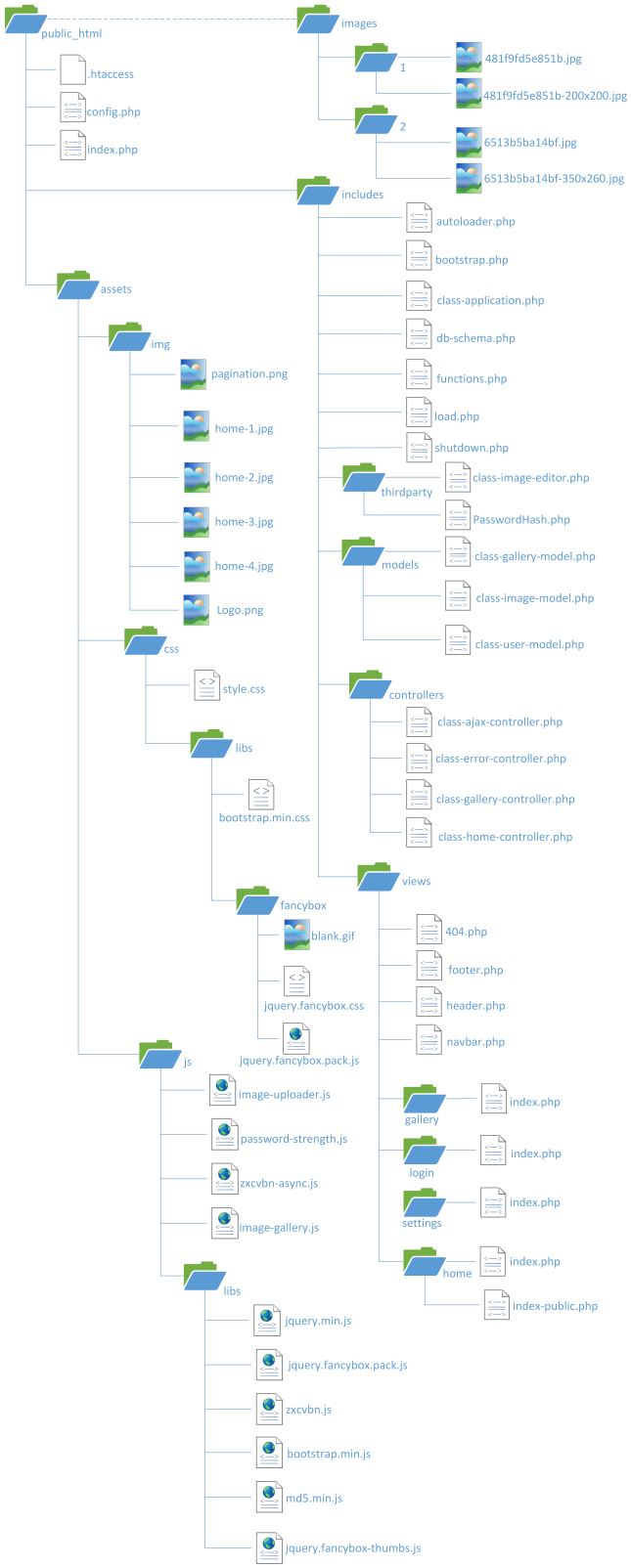
$app−>config−>segment_offset = 1; if you have used a subdirectory, otherwise leave it as it is.

7. At the bottom of config.php you will find four database related settings. Please fill those settings accordingly to your database credentials. Eg. $app−>config−>db−>name = ''; should be named after the database you created at step no. 2. The last setting doesn't usually need any changes, leave it as it is if you are unsure. Don't forget to save the changes made.

8. Visit the URL that you have previously configured as the main location.

9. Let yourself guide through the automated installation process.

5 File structure



5.1 Where is what?

5.1.1 AJAX, jQuery and JSON

- /assets/js/image-gallery.js
- /assets/js/image-uploader.js

5.1.2 MySQL Connection

- /includes/class-database.php

5.1.3 User Login

- /includes/class-user-manager.php
- /includes/controllers/class-login-manager.php

5.1.4 Lightbox

- /assets/js/image-gallery.js
- /includes/views/gallery/index.php