Exercise Sheet 4

(PHP preparations and JSON)

Submission until: 01./04.12.2024 23:59h **Presentation:** 02./05.12.2024

Introduction

This exercise sheet 4 focuses primarily on the first server-side PHP implementations and the JSON format. Therefore, a web server with a configured PHP execution environment is required.

We recommend you to use **XAMPP**. This tool provides (among other things) **Apache** as a web server, the execution environment for **PHP**, and the database management system **MariaDB**.

Task 1 (Preparing the Environment)

Download (https://www.apachefriends.org/de/download.html), install and start XAMPP, before you begin to work on this task sheet.

After the installation of XAMPP, open the **htdocs** folder (you will find a button for this in the GUI of XAMPP) and create a new subfolder inside this folder for your project, e.g., **myWebShop**.

You can copy all necessary files from the previous task sheets inside the newly created folder. Finally, you must start Apache to let the server run under: http://localhost/myWebShop/

Task 2 (Updating Files)

In this task you should update some files of your web-shop and prepare them for dynamic content generation based on the server-side page generation.

All components of the application will be preprocessed using PHP at the server. To do this, all HTML files must be renamed and converted to a file having a **.php** extension. It means that the links in your files will probably no longer work. You must update them to the new file names with the .php extension as well.

Think about the necessity of your files. You will not need individual pages for every product as dynamic code allows you to add changing content to one (template) page based on the current context.

Create a basic structure and add one or more PHP blocks to the body element as follows:

```
<!-- ... -->
<body>
   <?php
     // your PHP code
   ?>
</body>
<!-- ... -->
```

Start by updating the **index.php** and **about.php** files first. As we don't use a database at this point, you can simply use variables in the code containing the text for the both simple pages (or if you like, you can save the text optionally in a file that you can load in the relevant areas of the page).

Example: `$text1 = "This is my first text in a variable.";`

Task 3 (JSON Data Files)

In this task you should prepare appropriate JSON files containing all important information for the presentation of products, i.e.,

- product-ID (pid),
- name,
- description,
- path to an image,
- category,
- subcategory,
- price

You can also add further product data, if necessary. The JSON file can be structured similarly to the following code snippet, but another structure can also be used for the JSON files, e.g., you can create one file for each category.

```
{
    "product": [
        {
            "pid": 1,
            "name": "Sneaker 2452 Air",
            "description": "This shoes are very comfortable!",
            "imagepath": "./images/sn_2452_air.jpg",
            "price": 79.99",
            "category": "shoes",
            "subcategory": "sneaker"
        },
        {
            "pid": 2,
            "name": "Sneaker Max 20",
            "description": "This shoes are very comfortable!",
            "imagepath": "./images/sn_max_20.jpg",
            "price": 99.99",
            "category": "shoes",
            "subcategory": "sneaker"
        },
        {
            "pid": 3,
            "name": "Boot 3211",
            "description": "This shoes are very comfortable!",
            "imagepath": "./images/bt_3211.jpg",
            "price": 169.99",
            "category": "shoes",
            "subcategory": "boots"
        }
    ]
}
```

Task 4 (Category and Product Page)

Now you can update your product category page and the product detail presentation page. First, find a solution to read the previously created JSON files in PHP and to parse the information.

Once you imported the data, you can use the information to fill the **product.php** (template-)page with the loaded texts. You can use the path to the image to import the product's picture.

Use queries to pass variables via the URL. For example, you can use the following example to present the product with the ID 1234.

http://localhost/myWebShop/product.php?pid=1234

In PHP, you can query the information via the magic variable **$_GET** (or **$_REQUEST**):

```
echo $_GET["pid"];
```

Now you can implement the categoryList pages. By clicking on a product, the product's detail page will be shown by using appropriate queries.

Consider the error cases! You should check the variables and define an appropriate behavior. Use the following snippet in a correct context for this task:

```
if(isset($_GET["pid"])) {

   if(empty($_GET["pid"])) {

      echo "No value fo the parameter!";
   }

} else {
   echo "Parameter is missing!";
}
```

Check the behavior of the following URLs:

http://localhost/myWebShop/product.php?pid=
http://localhost/myWebShop/product.php

Finally, you can use two variables in a common link in order to present two products on the same page, e.g.,: http://localhost/myWebShop/product.php?pid1=1234&pid2=2222

Task 5 (Ideas for Shopping Cart Implementation)

In this last task, you should find an approach to implement our missing page for the shopping cart and present it to the class. Therefore, you should research some best practice examples in the web. It is not necessary to implement your own solution at this stage.

Submission hints:

- Upload a solution **ZIP document** containing all solution files (i.e. html, css, js, and pdf).
- Please write all names of the team that participated in the solving process of this task in the PDF file.
- All team members must be present for a certificate in the following course. For a successful acceptance, the proposed solution must be explained by all team members.