Начинаем постепенно разрабатывать сайт, основу которого мы создали в первом модуле. На сайте есть форма для авторизации пользователя. В этом модуле мы реализуем обработку данных, отправляемых этой формой (логин и пароль), а после обработки выведем сообщение об успешности авторизации.

1. Веб-форма для авторизации не должна отображаться всегда. Она появляется только при переходе на отдельную страницу авторизации. При этом, хотя страница и будет считаться отдельной, на деле это будет всё та же главная страница. В вёрстке предусмотрена ссылка на страницу «Авторизация». Укажите в ней ссылку на главную страницу своего сайта с дополнительным GET-параметром:

|  |
| --- |
| login=yes |

1. На главной странице интегрируйте логику отображения формы авторизации: она должна выводиться на странице только при передаче GET-параметра login в адресной строке браузера в значении yes.
2. Данные, указанные пользователем в форме авторизации, должны отправляться методом POST на эту же страницу вашего сайта. Для этого необходимо в атрибуте action тега form указать корректную ссылку на текущую страницу с необходимыми GET-параметрами.
3. Если форма отправлена (проверяем условием if суперглобальный массив $\_POST на пустоту), то на сервере должна осуществляться проверка введенного логина и пароля с определёнными значениями (см. п. 5). Необходимо проверить, что и логин, и пароль совпадают одновременно (только в этом случае можно считать авторизацию успешной).
4. Создайте два вспомогательных файла: /data/users.php и /data/passwords.php. В первом файле создайте массив с логинами пользователей, а во втором — массив с паролями. При этом к логину из массива логинов подходит один соответствующий (под тем же индексным ключом) пароль из массива паролей. Первым значением в массиве логинов укажите admin, а первым паролем в массиве паролей — pass. Также добавьте ещё несколько произвольных пар типа «логин — пароль».
5. Если форма отправлена, подключите файлы с логинами и паролями на странице авторизации. Переменные, созданные в этих файлах, станут доступны в том месте, где они подключены. Проверьте, что данные, введённые пользователем, совпадают с какой-либо заведённой парой «логин — пароль».
6. При совпадении и логина, и пароля необходимо вывести на странице сообщение «Вы успешно авторизовались». Сообщение необходимо вынести в подключаемый файл /include/success\_message.php и вывести на страницу путём подключения этого файла в нужном месте на странице.
7. В случае несовпадения логина и/или пароля необходимо вывести рядом с формой авторизации сообщение «Неверный логин или пароль». Вынесите его в отдельный подключаемый файл /include/error\_message.php.
8. Кроме того, при неуспешной попытке авторизации необходимо вывести в поля формы данные, которые пользователь ввёл ранее. Для этого в атрибуты value тегов input нужно подставить данные, которые пришли из формы, или подставлять пустые строки, если данных нет. Вспоминаем про замечательный тернарный оператор и используем его.

**Советы и рекомендации**

1. В п. 1 указана не полная ссылка, которую нужно разместить в теге, а только её часть, содержащая GET-параметры. Остальную часть нужно достроить самостоятельно.
2. Для ссылок на сайте используйте относительные полные ссылки от корня сайта (они начинаются со слеша) — что в форме авторизации, что в ссылке на страницу авторизации.
3. Не путайте URL и пути к файлам. Помните, что пути для подключения файлов внутри PHP — это пути внутри файловой системы, которые отсчитываются от корня файловой системы. Для подключения файлов в коде нужно использовать путь от директории, где лежит корень сайта: $\_SERVER['DOCUMENT\_ROOT']. А пути к подключаемым файлам (файлы стилей, изображения и т. п.) или страницам на сайте — это уже URL, который используется браузером. Корень URL — это и есть корень сайта, который начинается со слеша.
4. Код авторизации и вся логика приложения должны быть выполнены до начала HTML-кода страницы. Создание переменных, присваивание им значений, код обработки формы — всё должно быть там.
5. Создавайте вспомогательные переменные, если результат каких-либо условий работы кода требуется сохранить.
6. Лучше создать две разные boolean-переменные для обработки трёх различных ситуаций (форма не отправлена; форма отправлена, авторизация успешна; форма отправлена, авторизация не успешна), чем одну переменную, имеющую три состояния.
7. Никогда не помещайте теги в PHP-переменные и не выводите их с помощью оператора echo. Когда вам нужно вывести HTML, закрывайте PHP-код, выводите HTML, затем снова открывайте PHP и продолжайте код.
8. Для реализации задачи поиска элемента в массиве есть готовая функция в PHP, которую можно найти в документации.

***Критерии оценки***

1. Стили оформления кода psr-1 и psr-12 должны быть соблюдены.
2. Форма авторизации не должна отображаться при заходе на главную страницу.
3. При нажатии на ссылку «Авторизация» должна появиться форма авторизации.
4. Исполняемый PHP-код должен быть расположен до начала HTML-кода страницы.
5. Должна быть реализована проверка на пустоту массива POST.
6. При отправке формы, если она содержит корректные логин и пароль, вместо формы авторизации должно быть выведено сообщение об успешной авторизации, находящееся в файле /include/success\_message.php.
7. Если же значения не совпадают, над формой авторизации должно быть выведено сообщение об ошибке, находящееся в файле /include/error\_message.php.
8. Поиск логина должен быть осуществлён путём применения специальной подходящей функции для поиска элемента в массиве.
9. Функция должна быть применена только один раз, всю остальную логику проверки нужно осуществить на основе полученного результата работы этой функции.