

ЛАБОРАТОРНАЯ РАБОТА №4

SESSION. BOOTSTRAP

Цель работы: получить навык разработки сайтов с использованием языка программирования PHP

Задачи:

1. Разработать приложение на PHP, использующий сессии для хранения информации о текущем пользователе.
2. Реализовать динамическую загрузку данных при помощи AJAX.
3. Реализовать веб-интерфейс приложения с помощью CSS-фреймворка Bootstrap.

Результатами работы являются:

1. Разработанный сайт, содержащий PHP, JS и CSS файлы.
2. Подготовленный отчет.

СЕССИИ

Протокол HTTP(S) не поддерживает сессионность, т.е. каждый приходящий запрос на сервер рассматривается как изолированный, вне какого-то контекста. Иногда возникает необходимость связать ряд запросов в одну цепочку, например, запросы одного пользователя, для того чтобы разграничить права доступа и обеспечить отображение корректной информации. Для достижения такой цели необходимо использовать дополнительные механизмы.

Cookie (куки, печенки) — простые текстовые файлы на клиентском компьютере, по файлу для каждого домена (некоторые браузеры используют для хранения SQLite базу данных), при этом браузер накладывает ограничение на количество записей и размер хранимых данных (для большинства браузеров это 4096 байт). Для установки cookie с помощью PHP необходимо воспользоваться функцией:

```
int setcookie (string name [, string value [, int expire  
[, string path [, string domain [, int secure]]]])
```

Любые cookies, отправленные серверу браузером клиента, будут автоматически включены в суперглобальный массив `$_COOKIE` для обращения к ним со стороны сервера.

Чтобы удалить cookie достаточно в качестве срока действия указать какое-либо время в прошлом.

На основе cookies работает механизм поддержания сессий: при авторизации пользователя, сервер генерирует и запоминает уникальный ключ — идентификатор сессии, и сообщает его браузеру; браузер сохраняет этот ключ в cookie, и при каждом последующем запросе, его отправляет; проверяя по словарю значение полученной cookie, сервер может восстановить хранимое на сервере в отдельном файле содержимое сессии текущего пользователя.

В PHP уже реализован данный механизм: данные между запросами сохраняются в суперглобальном массиве `$_SESSION`. В тот момент, когда посетитель получает доступ к сайту, PHP проверяет автоматически (если `session.auto_start` установлено в 1) или по запросу

(явным образом через вызов `session_start()`), был ли определенный идентификатор сессии послан вместе с запросом. Если это так, восстанавливается сохраненное ранее окружение.

Пример работы с сессией в PHP:

```
//index.php
<?php header("Content-Type: text/html;
    charset=utf-8");
    session_start();

    if (!isset($_SESSION['count'])) {
        echo "Открытие сессии";
        $_SESSION['count'] = 0;
    } else {
        echo "Счетчик = ".$_SESSION['count'];
        $_SESSION['count']++;
    }
?>

<form action="destroy.php" method="post">
    <input type="submit" value="Убить сессию" />
</form>

//destroy.php
<?php header("Content-Type: text/html;
    charset=utf-8");
    session_start();
    session_destroy();
?>

<a href="test.php">Назад</a>
```

Также при работе с сессиями можно использовать дополнительные функции:

- `session_cache_expire` — вернуть текущее время жизни кеша
- `session_cache_limiter` — получить и/или установить текущий режим кеширования

- `session_create_id` — создаёт новый идентификатор сессии
- `session_decode` — декодирует данные сессии из закодированной строки сессии
- `session_destroy` — уничтожает все данные сессии
- `session_encode` — кодирует данные текущей сессии в формате строки сессии
- `session_gc` — выполняет сборку мусора данных сессии
- `session_get_cookie_params` — возвращает параметры cookie сессии
- `session_id` — получает и/или устанавливает идентификатор текущей сессии
- `session_is_registered` — определяет, зарегистрирована ли глобальная переменная в сессии
- `session_module_name` — возвращает и/или устанавливает модуль текущей сессии
- `session_name` — получить или установить имя текущей сессии
- `session_regenerate_id` — генерирует и обновляет идентификатор текущей сессии
- `session_register_shutdown` — функция завершения сессии

AJAX

AJAX — это аббревиатура, которая означает Asynchronous Javascript and XML.

В классической модели веб-приложения пользователь заходит на веб-страницу, совершает какое-то действие, после чего браузер формирует и отправляет запрос серверу. В ответ сервер генерирует совершенно новую веб-страницу и отправляет её браузеру и т. д., после чего браузер полностью перезагружает всю страницу.

При использовании AJAX пользователь заходит на веб-страницу и по совершению какого-то действия скрипт (на языке JavaScript) определяет, какая информация необходима для обновления страницы, браузер отправляет соответствующий запрос на сервер. Сервер

возвращает только ту часть документа, на которую пришёл запрос. Скрипт вносит изменения с учётом полученной информации (без полной перезагрузки страницы), например, добавляет некоторые элементы в [DOM](#).

Пример AJAX запроса с помощью нативного интерфейса fetch (fetch возвращает промис):

```
const response = await fetch(url, {
  method: 'POST', // *GET, POST, PUT, DELETE, etc.
  mode: 'cors', // no-cors, *cors, same-origin
  cache: 'no-cache', // *default, no-cache, reload,
    // force-cache, only-if-cached
  credentials: 'same-origin', // include,
    // *same-origin, omit
  headers: {
    'Content-Type': 'application/json'
  },
  redirect: 'follow', // manual, *follow, error
  referrerPolicy: 'no-referrer', // no-referrer,
    // *client
  body: JSON.stringify(data)
})
  .catch((error) => {
    console.log(error)
  });

if (response.ok) {
  return response.json();
}
```

BOOTSTRAP

Bootstrap — это открытый и бесплатный CSS фреймворк (конечно, не единственный в мире), который используется веб-разработчиками для быстрой верстки адаптивных дизайнов сайтов и веб-приложений.

В CSS Bootstrap содержит набор готовых правил для элементов различных классов: все классы имеют единую стилистику и приятную, легко изменяемую, цветовую схему (<https://getbootstrap.com/docs/5.2/components>).

Для ряда элементов, обладающих некоторым поведением, таких как выпадающие списки или radio-buttons, помимо css файлов необходимы соответствующие js-файлы.

Используя набор готовых правил можно существенно ускорить разработку, не отвлекаясь на дизайн, однако, нужно понимать, что на выходе получится шаблонный интерфейс, к тому, скорее всего с набором лишних зависимостей.

Кроме готового «дизайна» Bootstrap позволяет создавать адаптивную верстку с помощью специальной системы сеток.

Система сеток Bootstrap использует контейнеры, ряды и колонки, чтобы удобно располагать содержимое.

.container является корневым блоком сетки в Bootstrap, то есть располагается на внешнем уровне.

```
<div class="container">
  <div class="row">
    <div class="col">
      Этот текст внутри сетки Bootstrap
    </div>
  </div>
</div>
```

Контейнер может показаться избыточным элементом, но это не так. Он определяет ширину макета и выравнивает его по горизонтали в области просмотра. Кроме того, контейнер нужен, чтобы контролировать отрицательные внешние отступы рядов.

Название "ряд" (row) часто вводит в заблуждение и скрывает настоящее назначение рядов в сетке Bootstrap. Оно вызывает ошибочное представление о горизонтальной линии, однако колонки внутри ряда не всегда располагаются в одну строку. Их положение может меняться. Например, на маленьких экранах они могут выстраиваться друг под другом, а на больших – рядом. Лучше всего думать о ряде, как о родительском элементе для группы колонок.

Колонки нужны для деления области просмотра по горизонтали, при этом в одном ряду могут быть столбцы разной ширины. Их размер

может изменяться в зависимости от некоторых факторов. Пространство между колонками называется "желоб" (gutter). Родительский элемент делится на 12 колонок (Рис. 4.1).

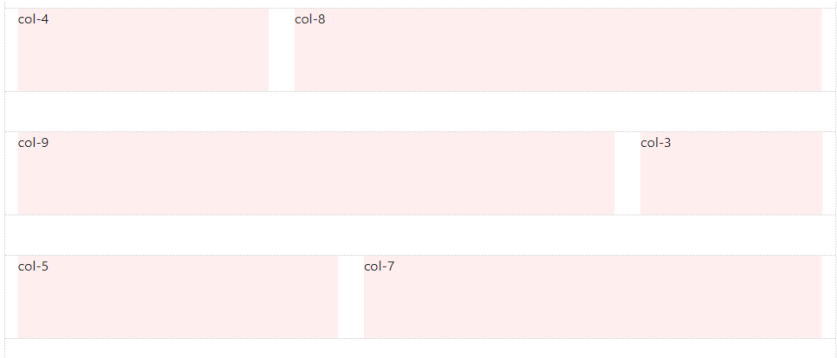


Рис. 4.1. Использование колонок в Bootstrap

Ширина одной и той же колонки, а также общая структура сетки может меняться для разных областей просмотра. Изменениями можно управлять с помощью специальных классов.

Фреймворк определяет 5 уровней адаптивности (брейкпоинтов), которые основаны на ширине области просмотра (Рис. 4.2).






					
	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Максимальная ширина контейнера	None (auto)	540px	720px	960px	1140px
Префикс класса	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
Число колонок	12				
Ширина отступа	30px (15px с каждой стороны столбца)				
Может быть вложенным	Да				
Упорядочивание колонок	Да				

Рис. 4.2. Классы адаптивных колонок

Большие брейкпоинты переопределяют меньшие: `xs(default) > sm > md > lg > xl`. Таким образом, класс `.col-sm-6` на самом деле означает, что ширина колонки будет составлять 50% на всех экранах размера `sm` и больше. Но класс `.col-lg-4` может переопределить это правило для больших и супербольших областей видимости, и на таких мониторах колонка будет занимать 33,3% ширины.

Если требуется установить одинаковую ширину столбцов на всех уровнях, достаточно явно указать ее для наименьшего.

Если не указывать размер колонок, то ряд равномерно поделится на все перечисленные столбцы. Для задания размера колонки по содержимому можно использовать классы `.col-{size}-auto`.

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Разработать сайт тематики, согласно полученному варианту.

Сайт должен требовать авторизации пользователей. Информация о пользователях хранится в отдельном файле (возможно использование формата xml или json). Должна быть предусмотрена регистрация новых пользователей (с сохранением в файл). Вся другая информация должна быть доступна только после авторизации.

На сайте должна быть отображена таблица (элемент bootstrap-table), содержащая информацию, хранимую на веб-сервере в отдельном текстовом файле (возможно использование формата xml или json). На сайте должна быть предусмотрена возможность добавления новых данных (с сохранением в файл).

Загрузка и передача данных должна осуществляться с помощью AJAX.

Вся валидация данных должна проводиться на стороне клиента без использования дополнительных библиотек.

ВАРИАНТЫ ЗАДАНИЙ

1. Сайт школы
2. Сайт фитнес-центра
3. Новостной сайт
4. Метеорологический сайт
5. Сайт музыкального исполнителя
6. Сайт кинотеатра
7. Сайт отеля
8. Сайт аэропорта
9. Сайт телеканала
10. Сайт радиостанции
11. Сайт автосалона
12. Сайт ресторана
13. Сайт университета
14. Сайт библиотеки

15. Сайт театра
16. Сайт ветеринарной клиники
17. Сайт туристической фирмы
18. Сайт агентства по продаже недвижимости
19. Сайт букмекерской компании
20. Сайт биржи криптовалют

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Опишите механизм работы cookies.
2. Опишите механизм работы сессии.
3. Приведите способ удаления cookie.
4. Дайте определение AJAX.
5. Раскройте, в чем преимущество использования AJAX.
6. Приведите пример использования нативного интерфейса fetch.
7. Раскройте, в чем преимущества и недостатки использования Bootstrap.
8. Опишите механизм создания адаптивной верстки с помощью Bootstrap.

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 5 часов: 4 часа на выполнение работы, 1 час на подготовку и защиту отчета по лабораторной работе.

Структура отчета: титульный лист, цель и задачи, формулировка задания (вариант), этапы выполнения работы, исходный код разработанного сайта, результаты выполнения работы (скриншоты), выводы.