

Министерство науки и высшего образования Российской Федерации
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Р. В. ПЕТРОВ, Д. В. ВАГИН

СОЗДАНИЕ СОВРЕМЕННЫХ КРОССПЛАТФОРМЕННЫХ ПРИЛОЖЕНИЙ НА ОСНОВЕ WEB-ТЕХНОЛОГИЙ

Утверждено
Редакционно-издательским советом университета
в качестве учебного пособия

НОВОСИБИРСК
2024

УДК 004.738.12(075.8)
ПЗ05

Рецензенты:

канд. техн. наук, заведующий лабораторией обработки и интерпретации данных АО «ЕМ-разведка» *А.В. Чернышев*
д-р техн. наук, профессор *М.Э. Рояк*

Работа подготовлена на кафедре прикладной математики НГТУ

Петров Р. В.

ПЗ05 Создание современных кроссплатформенных приложений на основе web-технологий : учебное пособие / Р. В. Петров, Д. В. Вагин. – Новосибирск : Изд-во НГТУ, 2024. – 64 с.

ISBN 978-5-7782-5240-0

В учебном пособии сначала дается теоретическая часть, а затем посредством несложных заданий предлагается изучить основные аспекты создания веб-сайта и применить полученные знания при создании собственной веб-программы, реализовав отсутствующий в CMS модуль. Также в учебном пособии даны примеры решения часто встречающихся задач.

УДК 004.738.12(075.8)

ISBN 978-5-7782-5240-0

© Петров Р. В., Вагин Д. В., 2024
© Новосибирский государственный
технический университет, 2024

ВВЕДЕНИЕ

О ВЕБ-РАЗРАБОТКЕ

В текущий момент веб-приложения очень популярны. Такая популярность связана с необходимостью реализации различных сервисов и программ, рассчитанных на широкий круг пользователей, которые не будут устанавливать себе на устройство (компьютер, телефон, телевизор и др.) какие-то внешние программы, но при этом заинтересованы в использовании вашего приложения. В этой ситуации часто используемым вариантом является создание прикладной программы на базе веб-технологий, которая решает поставленную задачу, а для ее использования достаточно браузера, который есть у всех. Поэтому наиболее часто встречающийся формат веб-приложения – это веб-сайт, или просто сайт. Альтернатива – это API веб-сервиса или мобильное приложение, которое также получает данные с сервера по веб-протоколам.

В данном учебном пособии будет преимущественно обсуждаться тема сайтов, но читателю не составит труда обобщить все изложенные здесь материалы на более широкий спектр веб-приложений.

СОВРЕМЕННОЕ СОСТОЯНИЕ В ОБЛАСТИ ВЕБ-РАЗРАБОТКИ

В настоящее время при веб-разработке применяются различные методики, языки программирования и технологии. В основном разница в применяемых технологиях зависит от размера, длительности и предполагаемой посещаемости проекта, а также от того, кто является конкретным заказчиком и исполнителем проекта.

Также следует отметить, что реальные проекты разрабатываются, как правило, не на «голом» языке программирования, а на фреймворке

(framework) или CMS-системе (Content Management Systems, система управления контентом).

С 2019 г., момента выхода предыдущей версии учебного пособия, возросла популярность Python как серверного языка программирования и Django, как фреймворка на этом языке. Также возросла популярность разработки веб-приложений на Typescript/React. Вместе с тем PHP, рассматриваемый в качестве основного языка в данном учебном пособии, по-прежнему является одним из самых популярных языков веб-разработки как в России, так и в мире, активно развивается, и по скорости освоения и простоте создания первых проектов является, на наш взгляд, хорошим стартом для изучения основ веб-разработки.

При этом, многие работодатели в своих вакансиях сразу требуют наличия опыта разработки под конкретный фреймворк. Например, для PHP такими фреймворками являются Laravel и Symfony. Однако, для работы в любом фреймворке всё равно нужно знать основы языка и в целом основы того, как работает веб-приложение.

Также изменились подходы к созданию сайтов. В 2019 г. и ранее было популярно создание собственных корпоративных сайтов, интернет-магазинов и др. В настоящее время хорошей альтернативой интернет-магазину является маркетплейс, альтернативой сайту-визитке – страница в соцсетях, либо сайт, собранный на SaaS сервисе (Tilda, Битрикс24.Сайты и т.д.), что не требует серверного программирования. Многие государственные организации запустили собственные сервисы для создания сайтов подведомственных организаций (так сделано, например, в Новосибирской области). Всё это уменьшает число мелких заказчиков и увеличивает запросы от внутренних ИТ-отделов компаний и организаций.

Нельзя забывать и про то, что с каждым годом всё большее значение приобретает безопасность веб-приложений. Поэтому в данной версии учебного пособия мы отдельно будем останавливаться на этих аспектах.

ОСНОВНАЯ ЦЕЛЬ КУРСА

Современные фреймворки и платформы разработки часто «скрывают» от программиста внутренние механизмы работы веб-приложения. Знание таких внутренних механизмов помогает качественно разрабатывать приложения на современных фреймворках и понимать, почему разрабатываемая система ведёт себя именно так.

Основная цель данного курса для студента – за несколько практических заданий (работ) пройти путь от статических страниц до разработки нового модуля на одном из языков веб-разработки или фреймворков; на практике изучить, как «внутри» веб-приложения происходит обработка внутренних данных и генерация контента. Для этого сформулированы следующие практические задания:

1. Основы интернета – HTML/CSS. То, с чего всё начиналось на frontend.

2. Современный CSS-фреймворк. На практике изучить историю развития веб-технологий на frontend.

3. Основы разработки на PHP и Python. С чего всё начиналось: без использования фреймворков сделать страницы сайта, обработку адресов, обращение к базе данных, сохранение и показ информации, авторизацию пользователей

4. Подключение интерактива на frontend: знакомство с современными JavaScript-фреймворками, их использование на практике. Знакомство с тем, как backend генерирует frontend.

5. Использование современных CMS. После выполнения этой работы у студентов должно сформироваться понимание того, что современные системы в своей основе имеют те же элементы, которые использовались во 2 и 3 работе.

6. Кастомизация дизайна CMS. Цель этой работы – дать понимание того, как объекты CMS отображаются в дизайне.

7. Разработка собственного модуля на CMS или фреймворке. В результате выполнения этой работы студенты должны получить практические навыки написания собственных модулей.

Эти работы, по мнению авторов данного учебного пособия, позволяют не просто изучить возможности какой-то одной системы, а получить понимание того, как устроено веб-приложение изнутри и откуда взялись те или иные механизмы и требования при его разработке.

ТИПЫ ВЕБ-ПРОЕКТОВ

Можно выделить несколько типов проектов, отличающихся разными подходами.

1. Небольшие по посещаемости, нагрузке и объему обрабатываемых данных сайты (до 5000 просмотров/день). Для них преимуще-

ственно используются готовые CMS, готовые решения на базе CMS, а также SaaS решения для построения сайтов (Tilda, Битрикс24.Сайты и др). Во многих случаях для таких проектов применяется типовой (шаблонный) дизайн, но иногда разрабатывается и индивидуальный дизайн. Примеры таких проектов – 90 % всех сайтов в интернете.

2. Средние сайты. Посещаемость и/или объем обрабатываемых данных для такого типа сайтов могут быть уже достаточно велики для применения индивидуальных программных модулей или решений. Также требования отдела маркетинга могут генерировать необходимость отдельного программирования и доработки стандартной функциональности CMS. Как правило, такие сайты уже имеют посещаемость более 5000 просмотров/день (или более 500 посетителей/день). Чаще всего такие сайты разрабатываются на базе коробочных CMS с разработкой дополнительных модулей. Для таких проектов чаще всего применяется индивидуальный дизайн. Иногда для подобных проектов разрабатывается собственная CMS на базе некоторого фреймворка. Примеры таких проектов – средние интернет-магазины, региональные порталы, региональные СМИ и др.

3. Большие проекты. Как правило, это либо проекты с большой посещаемостью, либо проекты с большим объемом данных и требованиями к работе $24 \times 7 \times 365$. Для них характерна очень серьезная программная доработка CMS или фреймворка для решения стоящих перед проектом задач. Как правило, в таких проектах используется несколько технологий и несколько языков программирования. Примеры таких проектов – крупные интернет-магазины (Эльдорадо, Связной, ...), крупные СМИ и др.

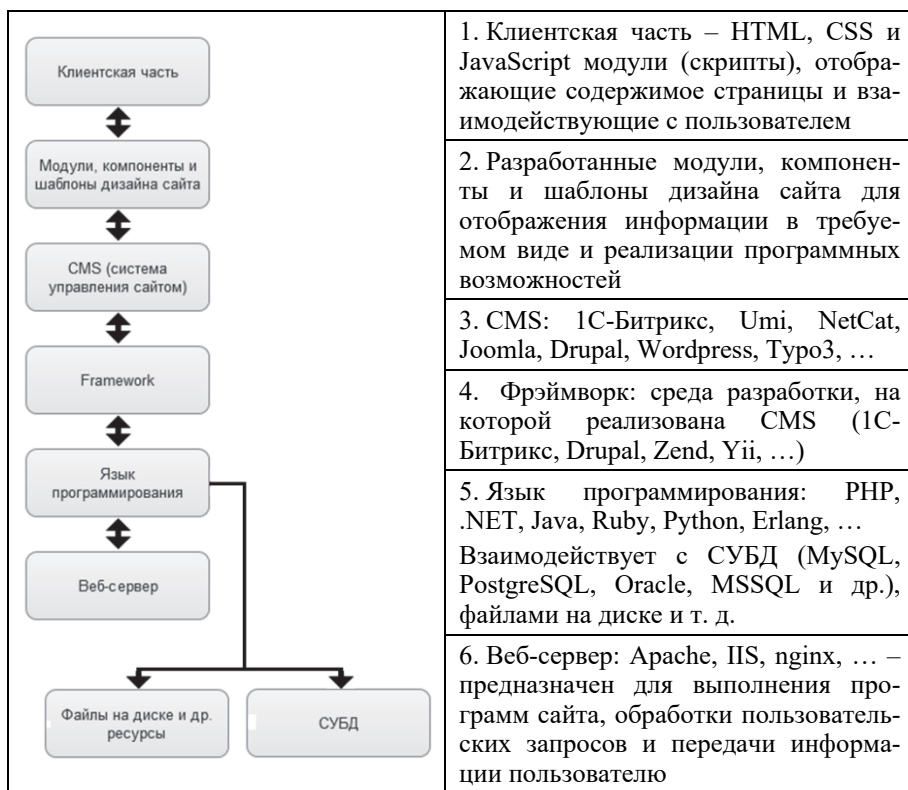
4. Enterprise решения, сервисы, SaaS проекты для построения более мелких веб-проектов. Характеризуются высокой нагрузкой, требованиями к работе $24 \times 7 \times 365$, долгим жизненным циклом, итерационной разработкой и большим числом пользователей с обязательным разделением по правам доступа. Для них характерна серьезная программная доработка CMS или фреймворка для решения стоящих перед проектом задач, объединение различных технологий и языков программирования. Примеры таких проектов – банковские сервисы, системы для коллективной работы, CRM системы (Битрикс24, АМО CRM), системы для поиска и бронирования авиабилетов, крупные мировые СМИ и сервисы (telegram, ВКонтакте, Одноклассники).

В настоящее время для большинства существующих и создаваемых веб-проектов существенный уклон делается в сторону:

- 1) соответствия сайта маркетинговым задачам компании;
- 2) соответствия дизайна сайта утвержденному дизайну;
- 3) удобства работы пользователя с сайтом;

4) и только в последнюю очередь – программирования сайта. Однако с развитием проекта и ростом посещаемости этот блок приобретает всё большее значение, поскольку при большой посещаемости или больших объемах данных оптимизация и правильно написанный программный код уже являются необходимым условием жизнеспособности проекта.

АРХИТЕКТУРА ВЕБ-ПРИЛОЖЕНИЯ



Такая архитектура приводит к тому, что создание сайта разбивается на несколько этапов. В том числе программирование сайта разбивается на клиентскую часть (**frontend**-программы, выполняющиеся в браузере клиента) и серверную часть (**backend**-программы, генерирующие страницы сайта).

Клиентская часть в настоящее время реализуется на языках JavaScript с использованием CSS3, HTML5. Для упрощения работы чаще всего используются уже готовые библиотеки JavaScript решений – JQuery и аналоги.

Для серверной части используются CMS и CMF (Content Management Framework), а также различные фреймворки для построения CMS и реализации модулей сайта.

О СТРУКТУРЕ ПОДАЧИ МАТЕРИАЛА В ДАННОМ УЧЕБНОМ ПОСОБИИ

Несмотря на то, что веб-приложение может использовать различные технологии, его основные составные части придуманы достаточно давно и эволюционируют, сохраняя основные принципы своей работы.

Поэтому в данном учебном пособии описано не только то, как разрабатывается веб-приложение и из каких частей оно состоит, но и отличия веб-приложений от классических Windows-приложений.

Составные части веб-приложения

1. Клиентская часть:

а) HTML и CSS, которые непосредственно отображают информацию в браузере пользователя;

б) код на javascript, который может изменять содержимое отображаемого HTML и CSS без перезагрузки страницы.

2. Серверная часть:

а) скрипты и программы, которые генерируют контент для клиентской части. Обращение к ним происходит по адресам с передачей необходимых параметров;

б) база данных;

в) статические файлы.

3. Внешние системы и сервисы, с которыми происходит взаимодействие.

В учебном пособии сначала дается теоретическая часть, а затем посредством несложных заданий предлагается изучить основные аспекты создания веб-сайта и применить полученные знания при создании собственного веб-приложения, реализовав отсутствующий в CMS модуль. Также в учебном пособии даны примеры решения некоторых часто встречающихся задач.

ЯЗЫКИ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ

Рассмотрим более подробно основные части веб-приложения и используемые языки программирования.

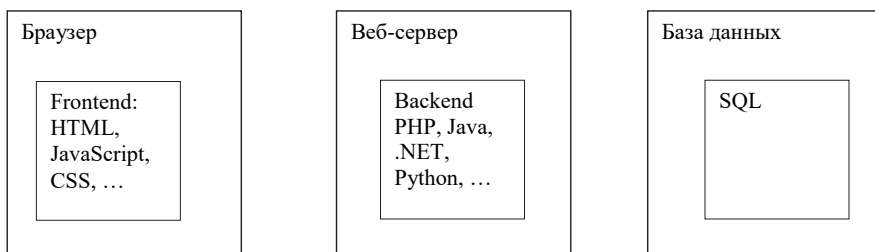


Рис. 1. Основные части веб-приложения

На рис. 1 схематично изображены основные части веб-приложения и их названия, а также используемые в них языки программирования.

FRONTEND

Frontend представляет из себя ту часть веб-приложения, которая непосредственно взаимодействует с пользователем. Обычно эта часть реализуется на HTML, CSS и JavaScript. Также возможны включения других технологий (WebGL и т. д.), но они используются намного реже.

Для frontend языков средой запуска (неким аналогом операционной системы) является браузер. Поэтому frontend часть веб-приложения работает до перезагрузки страницы (или перехода на новую страницу) либо до закрытия вкладки браузера.

При разработке frontend части сайта используются:

- **HTML.** Весь код в веб-приложении в итоге порождает HTML. Знание HTML обязательно для веб-разработчика;

- **CSS.** Сам по себе HTML довольно прост. Он реализует основные стили, но для создания хорошего интерфейса разработчики должны иметь опыт работы с CSS. Каскадные таблицы стилей предоставляют цвета, кнопки, подсветку и многое другое, что можно использовать для улучшения веб-страниц. Для облегчения работы с CSS придуманы такие языки, как Sass и LESS, также известные, как прекомпиляторы CSS. Они используются для написания более эффективного и управляемого кода CSS;

- **JavaScript.** JavaScript выполняется на компьютере пользователя. Перетаскивание, бесконечная прокрутка и видео, оживающее на веб-странице, могут быть запрограммированы с помощью JavaScript. Этот язык настолько популярен, что целые фреймворки построены исключительно для упрощения создания интерфейсов приложений. Такие фреймворки, как Angular, Ember, React и Backbone, используются для реализации сложных JavaScript-интерфейсов.

Изначально JavaScript был исключительно frontend языком, однако в настоящее время активно используется и серверный JavaScript.

BACKEND

Backend разработка отвечает за вычисления, бизнес-логику, взаимодействие с базами данных и производительность. Большая часть кода, необходимая для работы веб-приложения, будет выполняться на сервере. Внутренний код запускается на сервере, а клиенту (т. е. браузеру или приложению на JavaScript) отправляется HTML, PDF, либо любой другой набор данных, сгенерированных серверной программой. К backend языкам относятся следующие:

- **PHP** – один из самых популярных языков веб-разработки в мире. Благодаря простоте он завоевал множество поклонников, а его постоянное улучшение позволяет использовать его и дальше. Например, PHP версии 8 серьезно отличается от предыдущей версии и в плане возможностей, и в улучшении производительности, и в удалении из языка устаревших и неоптимальных конструкций.

- **Python** – популярный язык для веб-разработки, в том числе для работы с большими объемами данных и BigData, а также работой с нейросетями и др.

- **Java и .NET** – это платформы для разработки enterprise приложений (банковский сектор и т. д.), когда требуется реализация всех компонентов в единой среде.

- **Ruby и Ruby on Rails** – объектно-ориентированный язык и фреймворк для него.

- **NodeJS, Go, Erlang** – специализированные языки программирования для проектов с большим числом одновременных подключений.

Также существуют другие языки веб-разработки. По сути, любой язык может использоваться в качестве backend части, однако значительно эффективнее использовать языки специально предназначенные для веб-разработки.

СПЕЦИФИКА РАБОТЫ ВЕБ-ПРИЛОЖЕНИЙ

УПРОЩЕННАЯ СХЕМА РАБОТЫ ВЕБ-ПРИЛОЖЕНИЯ

До начала изучения веб-разработки читатель, скорее всего, имел дело только с классическими однопользовательскими приложениями либо с многопользовательскими клиент-серверными приложениями.

В случае однопользовательской программы, написанной, например, под Windows, в распоряжении программиста практически неограниченные ресурсы компьютера (оперативная память, процессорная мощность и время выполнения), возможность хранить информацию о пользователе, который работает в программе, и возможность хранить введенные пользователем (либо полученные программой) данные до момента завершения программы. При этом программа, по сути, работает на вашем компьютере и не делится на клиентскую и серверную часть.

В классическом клиент-серверном приложении ситуация очень похожа. В случае «толстого» клиента полученные клиентом данные обрабатываются на клиентской части, а в случае «тонкого» клиента – на сервере, при этом в классической схеме клиент-сервера одна из частей системы работает постоянно.

Стоит отметить, что при классической клиент-серверной архитектуре соединение между клиентом и сервером постоянно: устанавливается при старте программы и «держится», пока работа не завершена. Такая архитектура не позволяет обслуживать одновременно большое количество клиентов.

В веб-приложении ситуация отличается. Рассмотрим упрощенную схему работы веб-программы. Как правило, веб-приложение с точки зрения backend части представляет собой сайт (рис. 2).

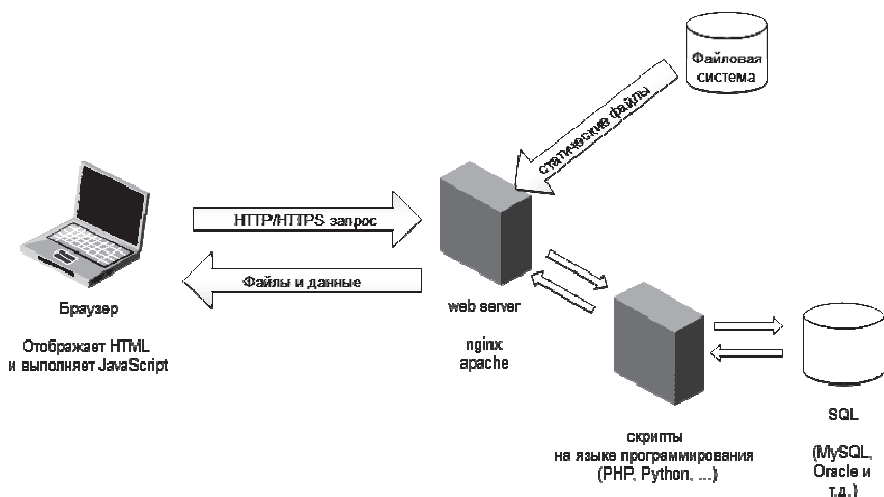


Рис. 2. Упрощенная схема работы сайта

1. При открытии страницы сайта браузер пользователя посылает запрос на веб-сервер с просьбой предоставить данные по запрошенному адресу.

2. Веб-сервер определяет, что делать с данным запросом и кто его будет исполнять. Наиболее частые варианты:

- а) прямая отдача файла;
- б) redirect (перенаправление) на другой адрес;
- в) запуск на выполнение программы на языке программирования и отдача клиенту результата выполнения программы.

3. В случае, если страница требует выполнения, в ней может быть ряд взаимодействий – с базой данных, внешними сервисами, файлами и т. д.

4. На frontend (в браузере клиента) также может выполняться программный код, который НЕ ЗАВИСИТ от программного кода на backend.

Таким образом, при разработке веб-приложения нужно учитывать, что у вас могут быть одновременно два приложения: на frontend и на backend. Backend-приложение работает на веб-сервере, вместо экрана у него (условно) браузер и оно ничего не знает о frontend.

Frontend приложение работает в браузере (каждая вкладка в браузере – аналог операционной системы) и оно ничего не знает о backend.

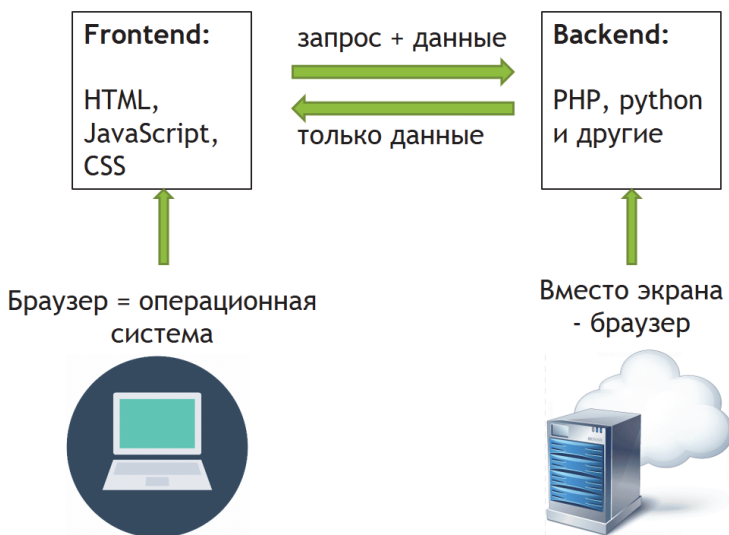


Рис. 3. Взаимодействие между backend и frontend

ЗАПУСК И ЖИЗНЕННЫЙ ЦИКЛ СЕРВЕРНОЙ ВЕБ-ПРОГРАММЫ ХРАНЕНИЕ И ПЕРЕДАЧА ДАННЫХ ВО ВРЕМЯ ВИЗИТА ПОЛЬЗОВАТЕЛЯ

Открытие новой страницы в большинстве случаев порождает запуск нового процесса на сервере. После передачи данных процесс завершается и соединение с сервером закрывается. Открытие этой же

страницы заново (перезагрузка страницы) приводит к запуску нового экземпляра программы и установке нового соединения.

Поэтому в классической ситуации при открытии страницы сайта веб-сервер не знает, было ли уже ранее обращение от данного пользователя (браузера) или нет. И необходимо как-то передавать информацию о том, что уже есть какие-то данные, которые нужно сохранять между переходами по сайту.

При этом необходимо защищаться от злоумышленников, которые хотели бы перехватить ваши данные (например, авторизоваться под вашим именем в личном кабинете интернет-банка).

Таким образом, на backend части сохранение данных при переходе пользователя между страницами сайта (и даже при обновлении одной и той же страницы сайта) представляет задачу передачи данных между двумя разными экземплярами выполняемой программы, один из которых к моменту запуска второго уже завершил свое выполнение.

В веб-разработке передача данных между страницами сайта организована на основе сессий.

Реализовано это так.

1. При первом заходе на страницу веб-сервер присваивает сессии пользователя уникальный идентификатор, создает на диске файл с указанным идентификатором и устанавливает в браузере посетителя cookie с данным идентификатором.

2. При открытии следующей страницы браузер посетителя отправляет на сервер идентификатор сессии, и по нему сервер находит файл с данными от предыдущей страницы сайта.

Естественно, что подобный механизм стал объектом пристального внимания злоумышленников. Если на заре веб-разработки идентификатор сессии передавался прямо в строке запроса браузера, то современные требования к сессиям – это привязка сессии к IP-адресу, передача идентификатора сессии в зашифрованном виде, хранение данных сессии не в файле, а в базе данных, периодическая смена идентификатора сессии и т.д. Всё это сделано для того, чтобы злоумышленники не могли перехватить данные посетителя сайта.

Также необходимо рассказать о хранении данных при создании приложения на JavaScript. В этом случае frontend-приложение представляет из себя аналог классического приложения, которое выполняется, пока не будет закрыто окно браузера или перезагружена страница, и может хранить данные в памяти приложения. При этом, если

происходит обмен с backend частью, даже по технологии AJAX, то в ход идет описанный выше механизм сессий, и можно говорить о том, что приложение на стороне клиента работает по клиент-серверной архитектуре и представляет из себя «толстого» клиента.

Следует отметить, что и в веб-разработке существует возможность работы с постоянным соединением – WebSocket, однако даже в этом случае механизм сессий остаётся актуальным.

Отметим также, что существует ряд backend языков, для которых открытие новой страницы не порождает запуска нового процесса. Но и для них справедливы вышеописанные вопросы безопасности данных.

SPA И MPA ПРИЛОЖЕНИЯ

Классическое веб-приложение работает так: в ответ на запрос пользователя на сервере генерируется HTML-код всей страницы сайта и отправляется клиенту. Если необходимо изменить какое-то поле или данные на странице, на сервер отправляется новый запрос, полностью генерируется новая HTML-страница, которая заново отображается в браузере. Такой подход хорошо работает, когда в браузере клиента нет сложной логики и интерактивных элементов. Данный тип веб-приложения называется MPA – Multi Page Application. Примеры – большинство интернет-магазинов.



Рис. 4. Иллюстрация отличия между MPA и SPA

Однако, с развитием браузеров и развитием JavaScript, появилась задача реализовать в браузере традиционные интерфейсы и интерактивные элементы (также как в desktop). Это можно сделать с помощью работы JavaScript в браузере. Для этого необходимо, чтобы вкладка браузера НЕ ПЕРЕЗАГРУЖАЛАСЬ при переходе по ссылкам, а вела себя как «аналог» обычного desktop приложения, т.е. постоянно работала, подгружая при необходимости данные из памяти приложения либо из внешнего источника по технологии клиент-сервер.

Такие приложения называются SPA – Single Page Application. Примеры – большинство мобильных приложений, соцсети, почтовые сервисы и т. д.

И у SPA, и у MPA есть свои плюсы и минусы – от SEO-продвижения до интерактивных возможностей, поэтому выбор варианта реализации определяется задачами. В данном учебном пособии рассматриваются MPA приложения.

СКОРОСТЬ ВЫПОЛНЕНИЯ

Рассмотрим классическую backend программу. Как правило, задачей веб-сервера является возможность одновременного обслуживания как можно большего числа посетителей.

Это означает, что большинство страниц сайта должно выполняться достаточно быстро, чтобы производительности системы хватило на большое число одновременных посещений.

В ряде исследований (например, <https://habr.com/ru/post/300210>) определено, что увеличение загрузки страницы даже на одну секунду приводит к снижению конверсии и увеличению показателей отказов на 2–6 %. Сейчас допустимым временем загрузки основного содержимого страницы сайта считается – не более трех секунд.

Учитывая затраты на передачу данных по сети, получается, что среднее время выполнения страницы сайта – от одной секунды для простых страниц и обычно не более 10 секунд (с поэтапной выдачей информации) – для сложных результатов поиска или фильтрации.

Если посмотреть на параметры предлагаемых сейчас тарифов хостинга, то можно увидеть, что обычный тариф хостинга не дает веб-программе более 60...200 секунд на выполнение. Ресурсы процессора и памяти тоже ограничены. Исключением являются дорогие тарифы,

в которых доступное время на выполнение программы может быть расширено до пяти минут и более.

Таким образом, задачей веб-сайта является быстро генерировать страницы сайта и быстро передавать их клиенту.

Возможность более долгого выполнения программы применяется в основном в служебных целях – например, при загрузке данных в интернет-магазин или для редкого открытия страницы со сложными отчетами.

В настоящее время скорость выполнения страниц сайта является одним из основных показателей при ранжировании сайта в поисковых системах, и написание кода программы так, чтобы она работала быстро – один из приоритетов при создании сайта.

Естественно, что только оптимизацией кода, мероприятия по ускорению сайта не ограничиваются. Применяются различные программно-аппаратные средства, начиная от оптимизации ПО веб-сервера и заканчивая различными технологиями кэширования и частичной загрузки страницы.

ЗАНИМАЕМАЯ ПАМЯТЬ

Аналогично вопросу с производительностью и скоростью выполнения программы у веб-сервера остро стоит вопрос с занимаемой памятью при выполнении веб-приложения.

Поясним на примере.

Классический процесс для веб-сервера Apache занимает от 32 до 64 МБ оперативной памяти. На каждое открытие страницы создается свой процесс. Процесс закрывается в тот момент, когда клиент закрывает соединение, или по окончании лимита времени на выполнение.

Таким образом, если генерация страницы занимает три секунды, и передача сгенерированного контента – еще три секунды, то для 100 одновременных соединений нам потребуется не менее 3,2...6,4 ГБ оперативной памяти. Нехитрый расчет показывает, что при суммарном времени работы одного процесса веб-сервера в шесть секунд одновременно сервер может выдержать не более 17 подключений.

В случае, если к сайту подключаются клиенты с медленным каналом связи, время передачи данных может увеличиться, и число одновременных подключений еще уменьшится.

Поэтому вопрос занимаемой памяти веб-программой решается сразу в нескольких направлениях.

1. Уменьшением объема памяти, занимаемым каждым конкретным процессом. Например, это может быть выбор другого веб-сервера или другого формата работы модуля для вашего языка, другой версии языка (например, переход на PHP7 дал уменьшение занимаемой памяти и ускорение программы до 20 %).

2. Уменьшением времени, которое каждый процесс тратит на работу. Помимо оптимизации кода, важный фактор – это время бездействия процесса, когда клиенту передаются данные или от клиента ожидается ответ на переданные данные. Это решается установкой акселератора (например, nginx), который быстро получает от веб-сервера данные, и затем отдает их клиенту уже так долго, как требуется клиенту. В этом случае время на передачу данных практически равно нулю, и число одновременно обслуживаемых посетителей возрастает.

Для очень высоконагруженных проектов применяются специализированные языки программирования.

БЕЗОПАСНОСТЬ

Одна из главных задач при разработке веб-сайтов и веб-приложений – это обеспечение безопасности.

Для этого сейчас применяются сразу несколько уровней защиты.

1. Использование HTTPS.

Ранее большинство сайтов работало по стандартному протоколу HTTP, в котором данные передавались в незашифрованном виде, и их можно было перехватить, имея доступ к каналу связи. В настоящий момент практически стандартом является использование протокола HTTPS (HTTP SECURE), в котором передача данных производится в зашифрованном виде.

2. Многоуровневая проверка доступа пользователя.

Если вопрос авторизации пользователя многие современные CMS берут на себя, то проверку наличия у пользователя прав на просмотр той или иной страницы или того или иного блока чаще всего веб-разработчик должен выполнять самостоятельно. Поэтому классиче-

ским вариантом является проверка в начале каждой страницы необходимых прав доступа и после этого принятие решения – отображать информацию на данной странице либо нет.

3. Контроль и очистка всех передаваемых данных пользователя.

При разработке веб-приложения вы должны привыкнуть к тому, что по умолчанию данным, пришедшим от пользователя, нельзя доверять. Это связано с тем, что подделать передаваемые данные достаточно несложно: если у злоумышленника есть доступ к компьютеру клиента, то он может заменить корректные данные, передаваемые браузером серверной части веб-программы, на некорректные. Поэтому вы всегда должны проверять входные данные, получаемые вашим веб-приложением.

Например,

- если вы ожидаете, что передаваемая от пользователя переменная должна иметь тип `integer` – явно приведите ее к данному типу;
- если в запросе к базе данных или в коде программы у используемых переменных есть недопустимые варианты (пусто, ноль, и т. д.) – обязательно проверьте и отсекийте данные варианты до начала выполнения основной части программы.

Для PHP это обычно реализуется следующим образом: поступившие от пользователя данные копируются из массива `$_REQUEST` (или `$_GET`, `$_POST`) в локальные переменные с приведением типа данных к необходимому и проверки значений на допустимость.

Например, `$id=(int) $_GET["id"]`

4. Защита от CSS, XSS, SQL -инъекций.

Дополнительным аспектом работы с данными является защита от XSS, CSS -инъекций и встраивания вредоносного кода в данные. Например, в `Bitrix framework` передаваемые пользователем данные проверяются на наличие подобных элементов, и в этом случае попавшие в программу данные модифицируются – например, в слово “`javascript`” добавляется пробел в середине слова. В случае, если передаваемые данные – это сообщение на форуме, человек прочитает его без труда, а XSS-инъекция уже не сработает. При этом у разработчика есть возможность получить немодифицированные данные и работать с исходным вариантом.

В качестве защиты от SQL-инъекций используются подготавливаемые запросы. Они позволяют передавать данные в поля СУБД без модификации, и при этом данные не влияют на SQL запрос. Использование подготавливаемых запросов в работе обязательно.

ВЗАИМОДЕЙСТВИЕ С СУБД

Отдельно следует остановиться на взаимодействии с СУБД. Самыми часто встречающимися СУБД для веб-разработки являются MySQL или PostgreSQL. Отметим, что множество справочной информации и учебников по работе с PHP и MySQL ориентирует читателя на устаревший драйвер работы с СУБД MySQL, который отсутствует в новых версиях PHP. Поэтому лучший вариант – это смотреть актуальную информацию о работе выбранного вами языка программирования с СУБД.

Второй часто встречающейся ошибкой является неоптимальное использование запросов к СУБД.

- Получение списка записей вместо одной записи.
- Получение всех полей данных вместо выбранных.
- Использование запросов к СУБД в цикле.
- Неиспользование всех возможностей СУБД по ускорению работы и снижению нагрузки на сайт.

Этот вопрос выходит за рамки нашего пособия, потому что мы не будем останавливаться на нем.

КЭШИРОВАНИЕ

Каким бы количеством вычислительных ресурсов вы не обладали, для быстрой работы сайта вам необходимо использовать кэширование часто встречавшейся ранее полученной информации. Для этого применяются различные технологии – от файлового кэша до специального сервера memcached.

Также в случае высоконагруженных проектов (например, мессенджеров) возможна ситуация, когда данные записываются и читаются из СУБД порциями и до момента записи/чтения хранятся в промежуточном буфере для быстрого доступа.

ЭТАПЫ РАЗРАБОТКИ ВЕБ-САЙТА

Первым этапом разработки сайта является получение от заказчика бизнес-цели сайта.

Например,

ваш заказчик – небольшая торговая компания, работающая в г. Новосибирске, продающая велосипеды и аксессуары к ним (шлемы, запчасти и т. д.). Общее число товаров не превышает 500 позиций. Для каждого товара важны вес, фотографии с разных ракурсов, текстовое описание и цена. Цены не зависят от категории покупателей и являются фиксированными. Оплата – только наличными курьеру. Ваша задача – разработать для вашего заказчика интернет-магазин.

Рассмотрим далее типовой процесс разработки сайта. Их в настоящее время существует два: классический – с полным проектированием и разработкой и ускоренный – когда приобретается некий готовый сайт или сайт собирается из готовых модулей, а затем дорабатывается.

ТИПОВОЙ ПРОЦЕСС РАЗРАБОТКИ САЙТА (КЛАССИЧЕСКИЙ)

1. Сбор требований.
2. Проектирование.
3. Прототипирование.
4. Разработка технического задания (ТЗ).
5. Определение дизайна.
6. Верстка, программирование клиентской части.
7. Интеграция дизайна и CMS, программирование серверной и клиентской части.
8. Интеграция с 1С/ERP.
9. Тестирование.
10. Обучение.
11. Наполнение.
12. Запуск.
13. Гарантийная поддержка.
14. Доработки и развитие.

Несмотря на то, что собственно программирование начинается только на седьмом этапе, привлечение программиста необходимо (либо менеджера проекта с соответствующей квалификацией) и на более ранних этапах для того, чтобы избежать ошибок и переделки уже выполненных этапов проекта из-за несогласованности форматов данных, способов обмена, сценариев работы пользователей и др.

Альтернативой классическому процессу разработки сайта является старт на готовом решении или на шаблоне – это позволяет сразу получить минимально работающий продукт, и затем точно улучшать его.

Рассмотрим более подробно каждый этап классической схемы разработки сайта.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Для поставленной цели по разработке сайта необходимо собрать требования и разработать техническое задание, которое включает следующее.

1. Цели и задачи сайта.
2. Предполагаемую структуру сайта.
3. Способ заполнения каталога товаров (вносится вручную прямо на сайте, загружается из учетной системы, смешанный вариант и т. д.).
4. Структуру данных каталога товара и другой динамической информации (новости, акции и т.д.), если это требуется.
5. Поведение пользователей при просмотре товара и его заказе.
6. Способы регистрации пользователей и необходимые поля.
7. Структуру заказа товара, формат хранения данных о пользователе, его заказах, адресах доставки и др.
8. Способы оплаты и доставки, формат обмена заказами с учетной системой (если требуется), формат предоставляемых покупателю документов (счет, счет-фактура, товарная накладная и др.).
9. Другие параметры.

При этом следует учитывать, что большинство CMS уже обладает возможностями для создания, редактирования, вывода каталога товаров и другой динамической информации, модулями для регистрации и авторизации пользователей, модулями для оформления, редактирования и других операций с заказами, модулями для проведения оплаты, возможностями для обмена данными с учетными системами и др. Поэтому при разработке ТЗ необходимо учитывать возможности той

CMS, которую вы планируете использовать для построения сайта, и максимально следовать идеологии, принятой в используемой CMS.

Часто используемый сейчас вариант – с помощью эксперта-аналитика выбрать подходящий готовый интернет-магазин (во многих CMS есть готовые наборы) или установить демо-версию, и только затем перейти к написанию технического задания и остальным этапам, чтобы учесть в ТЗ уже имеющиеся возможности CMS.

На этапе разработки ТЗ вам необходимо учитывать, что некоторые доработки могут потребовать значительного времени (и финансовых затрат) и при этом не представлять никакой финансовой пользы для заказчика. Если вы обладаете влиянием на проект, вы можете акцентировать на это внимание и сократить тем самым затраты на проект.

Чаще всего проектирование сайта идет параллельно с его прототипированием. Это связано с тем, что без графической визуализации заказчик и исполнитель, как правило, по-разному представляют себе конечный результат. Иногда вместо прототипирования сразу рисуется дизайн, но, как правило, это приводит к дополнительным затратам на разработку проекта из-за сложности внесения изменений.

Финальное техническое задание обычно появляется после окончания прототипирования, и дальнейшие правки в него уже относятся к этапу развития сайта.

РАЗРАБОТКА ПРОТОТИПОВ СТРАНИЦ

Для посетителей сайта подарков важно сразу увидеть ассортимент продукции, легко положить в корзину товар, посмотреть содержимое корзины и стоимость заказа и сделать заказ. Также очень важным является удобное и хорошо заметное расположение контактов (телефона), чтобы можно было уточнить заказ голосом и задать необходимые вопросы. Индивидуальность проекту должен придать фирменный стиль и логотип компании. В остальном возможно использование шаблонного дизайна. Необходимо предусмотреть заказ без регистрации и регистрацию и вход в личный кабинет для отслеживания статуса заказов.

При разработке прототипов большое внимание следует уделить проработанности каждого блока страницы. Если какой-то блок заявлен в прототипе (например, телефон магазина в шапке сайта), то этот блок должен содержать корректную информацию, позволяющую

в прототипе увидеть не только то, где она будет расположена, но и как она будет выглядеть и сколько места она будет занимать. Например, для блока «телефон» корректным является следующее представление: «тел. +7 495 123-45-67», и некорректным: «тел. Здесь будет телефон». Для блока новости обязательно указание даты новости, заголовка новости. Для товара обязательно указание корректного наименования товара. Для выполнения работ выберите в реальном веломагазине несколько товаров с коротким (два-три слова), длинным (более 10 слов) или длинными словами и средними названиями и укажите их на прототипе. При разработке прототипов следует учитывать размеры экранов, на которых будет отображаться сайт. Для экранов современных планшетов и компьютеров используется разрешение от 1024 до 5120 точек. Самые часто используемые разрешения для компьютеров – 1366 и 1920 точек, а для мобильных устройств возможны варианты горизонтального и вертикального отображения, и логическая ширина экрана начинается от 360 точек. На рис. 5 показан пример прототипа главной страницы сайта для отображения на мониторах компьютеров с корректным использованием всех элементов.

При прототипировании и последующей разработке дизайна важно учесть адаптивность, т.е. просмотр сайта на разных экранах. Как правило, для этого рисуется столько макетов, сколько основных разрешений вы планируете использовать, например:

- вариант для desktop с разрешением 1920;
- вариант для desktop с разрешением 1366;
- вариант для мобильных в вертикальной ориентации.

Затем определяется правило изменения блоков и их положения. Современные CSS-фреймворки позволяют упростить данный процесс, предлагая стандартные варианты сетки сайта и механизмов перестроения элементов сайта при уменьшении размера экрана.

ДИЗАЙН

После утверждения прототипов и утверждения технического задания наступает этап дизайна.

Этап дизайна – очень важный этап для создания сайта. Однако это не является предметом данного учебного курса, как и верстка полученного дизайна, поэтому выберем подходящий по структуре бесплатный

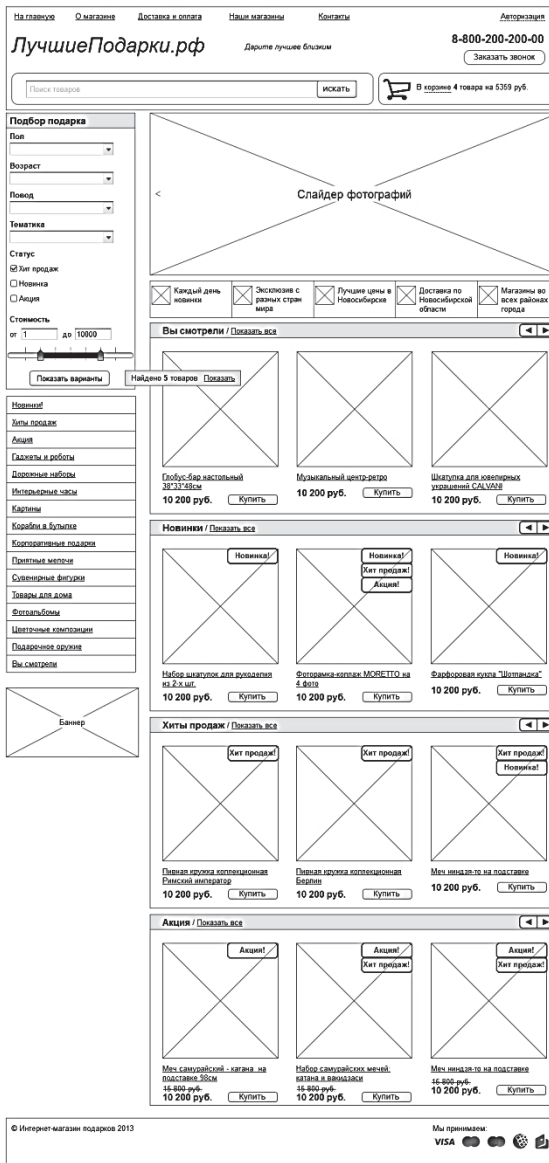


Рис. 5. Пример прототипа страницы сайта

шаблон из набора шаблонов 1С-Битрикс либо используем подходящий для нас вариант с сайта www.templatemonster.ru с доработкой дизайнером.

Данный подход не позволит получить качественный, с маркетинговой точки зрения, сайт, но позволит нам не заострять внимание на этапе дизайна, а перейти сразу к разработке и программированию.

HTML ВЕРСТКА

Если раньше HTML верстку сайта выполняли вручную, то сейчас широко используются CSS-фреймворки, например Bootstrap (<https://getbootstrap.com>), что позволяет существенно снизить затраты и ускорить процесс. Но при этом прототипы и дизайн также должны быть сделаны в концепции выбранного CSS-фреймворка.

ПОДКЛЮЧЕНИЕ FRONTEND СКРИПТОВ

Даже несложные сайты сейчас широко используют возможности JavaScript. Для удобства работы многие используют JavaScript-фреймворки – такие, как jQuery (www.jquery.com), которые позволяют более простым способом получить необходимые возможности в frontend части сайта.

РАЗРАБОТКА BACKEND ЧАСТИ

Как правило, backend часть разрабатывают на базе CMS. В этом случае оптимальный вариант – использовать для обучения учебные курсы производителя CMS.

Например, для 1С-Битрикс рекомендуются курсы на официальном сайте: <https://academy.1c-bitrix.ru/types/programming>.

Для других систем следует использовать учебные курсы от этих систем.

Отдельно стоит отметить базовые курсы по языкам веб-разработки. Очень часто общедоступными в интернете являются устаревшие курсы. Например, для разработки на PHP мы рекомендуем использовать официальную документацию именно с сайта PHP.NET, а не с зеркал. Обратите внимание, что на официальном сайте PHP документация доступна и на русском языке: <https://www.php.net/manual/ru>.

Связано это с тем, что после выпуска PHP7, ряд устаревших функций был отключен. Например, в большинстве источников для подключения к СУБД MySQL рекомендуется использовать библиотеку `mysql`, однако она отключена в PHP7, вместо нее используется современная библиотека `mysqli`, и т.д.

ТЕСТИРОВАНИЕ И ЗАПУСК ПРОЕКТА

После разработки веб-проекта его необходимо протестировать: как правило, не только корректную работу backend, но и корректное отображение в различных браузерах и на различных устройствах. Также необходимо протестировать на корректную обработку «неправильных» данных, т. е. на безопасность работы веб-сайта.

Важным блоком работ является опытно-промышленная эксплуатация и запуск проекта. Многие начинающие веб-разработчики не выделяют на это ресурсы, а между тем это существенная часть работы.

После этого проект переводится на рабочую площадку и наступает этап развития и сопровождения.

РАЗВИТИЕ ВЕБ-ПРОЕКТА

Современный подход к выпуску программных систем часто требует выпуска продукта с минимально работающей функциональностью и последующим его развитием. Поэтому для веб-приложения очень важна возможность развития и расширения функциональности.

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Мы предлагаем вам начать вхождение в мир веб-разработки с набора заданий, которые позволят вам пройти путь веб-разработки от простого HTML до разработки на фреймворке за несколько занятий.

Выполнение заданий производится на примере новостного сайта.

Последовательность заданий

1. Создание страниц на HTML и CSS.

2. Подключение CSS Framework для придания новостному сайту современного вида.
3. Создание новостного сайта на HTML, CSS, PHP, MySQL.
4. Подключение JavaScript и jQuery.
5. Создание новостного сайта на базе современной CMS (со встроенным дизайном):
 - а) 1С-Битрикс;
 - б) Wordpress.
6. Подключение CSS Framework (собственного дизайна) к разработанному новостному сайту.
7. Разработка собственного модуля для CMS/индивидуального модуля для сайта.

ОБЩЕЕ ЗАДАНИЕ НА ПРОЕКТ

На протяжении работ 1–6 мы будем делать новостной сайт. Новости – это классический пример для построения сайтов, веб-приложений и т. д., поскольку они достаточно просты в реализации, и обладают необходимыми свойствами для создания учебного проекта.

Итак, новостной сайт – это сайт, который публикует новости. Примеры таких сайтов – lenta.ru, rbc.ru и другие.

Новость обычно хранится в таблице базы данных. Минимально необходимый набор полей для новости такой:

- Id
- Дата/время
- Заголовок
- Анонс
- Полный текст новости
- Адрес картинки

Обратите внимание, что в веб-разработке не принято хранить картинки в базе данных. Для целей производительности и экономии ресурсов картинки хранятся в файловой системе, а в базе данных хранится путь к ним.

ПОСЛЕДОВАТЕЛЬНОСТЬ ДЕЙСТВИЙ ДЛЯ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ РАБОТ

УСТАНОВКА ВЕБ-СЕРВЕРА

Для разработки новостного сайта нам потребуется веб-сервер. Мы рекомендуем установить XAMPP:

<https://www.apachefriends.org/index.html>

Это готовая сборка веб-сервера, содержащая apache, php, mysql и другие пакеты.

В результате установки у вас появится папка C:\xampp.

Для запуска XAMPP необходимо запустить файл xampp-control.exe и затем запустить первые два пункта – веб-сервер и MySQL (рис. 6).

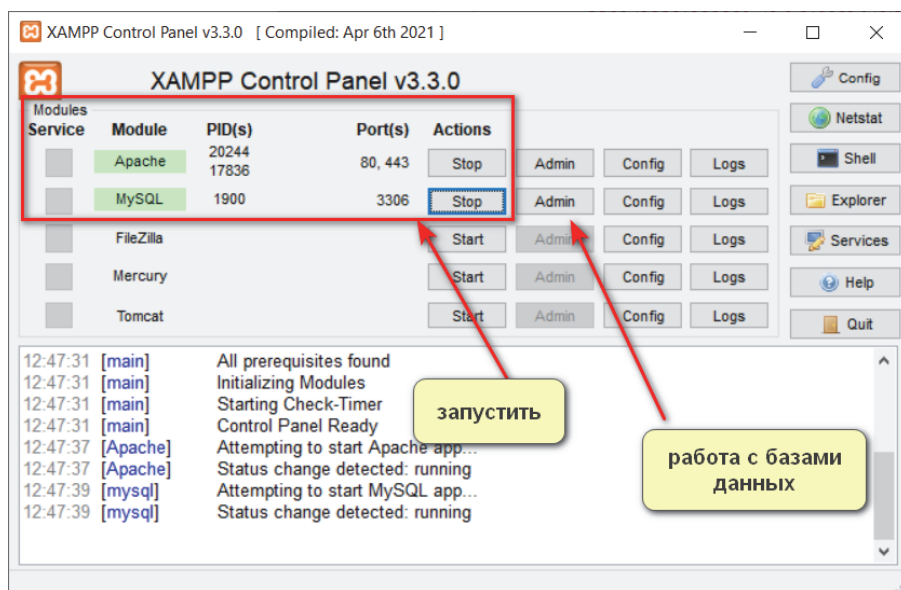


Рис. 6. Панель управления сервисом XAMPP

Далее все работы нужно делать в папке C:\xampp\htdocs. Хотя работы 1–2 не требуют наличия веб-сервера, а позволяют работать локально, мы рекомендуем сразу начинать работать правильно – через веб-сервер.

Отличие от открытия файла с локального диска в том, что файл будет обработан веб-сервером, и, если этот файл написан на backend языке программирования, будут выполнена программа, записанная в этом файле. При локальном открытии файла открывается его исходный код.

Папка C:\xampp\htdocs – это http://localhost. Соответственно, мы редактируем, например, C:\xampp\htdocs\index.php, а смотрим результат в браузере по адресу http://localhost/index.php.

Поскольку выше, чем localhost, в адресе «подняться» нельзя, появляется закономерное правило – нельзя в вашем сайте ссылаться на файлы выше корня сайта (C:\xampp\htdocs). При этом PHP может открывать и подключать файлы из любой локации, но результат должен быть выдан на экран в действующей программе.

Также, обращаем внимание на имена файлов. Принято стартовую страницу называть index.php, а другим файлам давать имена маленькими английскими буквами и цифрами без пробелов.

СОЗДАНИЕ СТРАНИЦ НА HTML И CSS

Цель данной работы – без использования фреймворка создать несколько страниц на HTML и CSS, которые затем будут использованы как шаблоны новостного сайта.

В простейшем случае нам будет нужно реализовать три HTML страницы, и затем запрограммировать их (в других работах).

1. Страница со списком новостей (3 новости + пагинация).
2. Страница с детальным текстом новости (3 страницы, по одной для каждой новости).
3. Страница с контактной информацией.

Критерии оценки:

- Реализованы страницы со списком новостей, новостью детально, контактами.
- У новостей есть дата, заголовок, текст анонса, картинка, детальный текст.
- Картинки не потеряли свои пропорции.
- CSS вынесен в отдельный файл.
- Текст новостей взят из реальных новостей.
- Сайт выглядит целостным, ничто никуда не разъезжается.
- Желательно: адаптивность.

Для создания HTML страниц можно использовать примеры с сайта htmlbook.ru, раздел «Сайт»: <http://htmlbook.ru/layout>.

Также на этом же сайте можно на русском языке получить обширную справочную информацию об HTML и CSS.

Если необходимо более детальное изучение HTML и CSS, можно изучить бесплатные курсы на <https://htmlacademy.ru>.

ПОДКЛЮЧЕНИЕ CSS FRAMEWORK ДЛЯ ПРИДАНИЯ НОВОСТНОМУ САЙТУ СОВРЕМЕННОГО ВИДА

На текущий момент мы с вами создали новостной сайт, однако по внешнему виду он не соответствует требованиям, предъявляемым к современным сайтам. Например, он скорее всего не адаптивный, т. е. не подготовлен для отображения на устройствах с разным размером экрана, не выстроен «по сетке» дизайна и т.д.

Для упрощения создания современных сайтов разработан ряд CSS-фреймворков, например Bootstrap (<https://getbootstrap.com>), что позволяет существенно снизить затраты и ускорить процесс.

Предлагаем подключить к нашему сайту Bootstrap и посмотреть, что получится. Для этого вы можете как самостоятельно сделать задание из работы 1 на bootstrap, так и выбрать и модифицировать любой бесплатный шаблон bootstrap, содержащий в себе заготовки для новостного сайта. Результат вас приятно удивит – с минимальными затратами времени вы создали современный по внешнему виду и поведению сайт!

Найти шаблоны и плагины для Bootstrap можно, например, на сайте <http://bootstraptema.ru>. Документация по созданию сайта на bootstrap размещена на официальном сайте <https://getbootstrap.com/docs>.

Также, вы можете выбрать любой другой современный CSS-фреймворк вместо bootstrap.

Задание на данную работу повторяет задание для работы 1, со следующими дополнениями:

- Все элементы, присутствующие на странице, имеют какое-то значение (т. е., если выбран готовый шаблон, то на сайте не осталось неиспользованных блоков шаблона).

- Вы сделали данные страницы по методологии выбранного CSS-фреймворка, а не подключили современный фреймворк к ранее сделанной работе 1.

СОЗДАНИЕ НОВОСТНОГО САЙТА НА HTML, CSS, PHP, MYSQL

После того как шаблоны трёх страниц сайта реализованы на HTML и CSS, можно сделать их динамическими – т.е. перенести на PHP и добавить получение данных из базы данных.

ОСНОВЫ PHP

Сначала рассмотрим основы создания сайта на PHP.

PHP – это, условно, интерпретируемый язык. Для запуска программы/сайта на PHP не требуется ее компиляция, достаточно открыть соответствующий адрес в браузере. Адрес, конечно, должен начинаться с http или https, т. е. интерпретатор PHP должен выполнить программу.

Когда PHP обрабатывает файл, он ищет открывающие и закрывающие теги, такие как `<?php` и `?>`, которые указывают PHP, когда начинать и заканчивать обработку кода между ними. Подобный способ обработки позволяет PHP внедряться во все виды различных документов, так как всё, что находится вне пары открывающих и закрывающих тегов, будет проигнорировано парсером PHP.

Если в одном и том же документе есть несколько конструкций между тэгами `<?php` и `?>`, то они, по сути, слепляет все эти конструкции в единый файл программы, а всё, что между ними просто выводит на экран.

Детальное описание типов данных и справочник языка, в том числе лучшие практики программирования:

<https://www.php.net/manual/ru/langref.php>.

Чуть позднее мы отдельно остановимся на ассоциативных массивах – конструкциях языка PHP, особенно удобных для работы с базами данных. Кроме того, в PHP огромное количество как встроенных, так и внешних функций и библиотек на все случаи жизни:

<https://www.php.net/manual/ru/funcref.php> – встроенные функции

<https://pear.php.net> – расширения

Пример программы на PHP

Разместите в файле index.php следующий код:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Тестируем PHP</title>
  </head>
  <body>
<?php
  $a=strftime("%d %b %Y", time() );
  ?>
    <p>Привет, мир! Сегодня <?=$a?></p>
  </body>
</html>
```

Таким образом, мы в программе на PHP получили текущую дату и время и отформатировали в нужном нам формате, а затем вывели переменную в требуемом месте.

ХОРОШИЙ И ПЛОХОЙ СТИЛЬ СОЧЕТАНИЯ PHP И HTML

Часто начинающие программисты при разработке на PHP смешивают HTML и PHP.

Чтобы этого не допускать, придумали различные варианты. В том числе разделение на «программу» и «шаблон», вынося логику в отдельный файл, а отображение результата – в отдельный файл шаблона. Для этого используются как внешние шаблонизаторы, так и встроенные средства языка PHP.

Независимо от того, используете ли вы разделение на программу и шаблон (например, в рамках концепции MVC (model-view-controller), мы рекомендуем даже в одном файле стараться не смешивать PHP и HTML. Хотя бы при выводе данных.

Обратите внимание на способ вывода переменной: в HTML код мы построили конструкцию `<?=$a?>`. Это позволяет удобно работать как frontend, так и backend специалисту. В отличие от варианта `printf` или

echo, в котором вы в кавычках указываете HTML, в рекомендуемом нами варианте вы «не ломаете» HTML код и делаете комфортной дальнейшую работу с приложением.

ЗАПРОСЫ К БАЗАМ ДАННЫХ И АССОЦИАТИВНЫЕ МАССИВЫ

Для работы с базами данных рекомендуем использовать подготавливаемые запросы, а результат сохранять в специальные конструкции языка PHP – ассоциативные массивы. Документация:

Краткое руководство: www.php.net/manual/ru/mysqli.quickstart.php.

Сохранение результата в ассоциативный массив:

<https://www.php.net/manual/ru/mysqli-result.fetch-assoc.php>.

СОЗДАНИЕ БАЗЫ ДАННЫХ И ТАБЛИЦ ДЛЯ НОВОСТНОГО САЙТА

Мы сделаем базу данных, которую назовём `mysite`, и в ней – одну таблицу `news`. Для этого в панели управления XAMPP нажмите кнопку «admin» напротив строки «MySQL» или наберите в браузере `http://localhost/phpmyadmin`. Далее выберите «Создать БД», укажите имя и выберите кодировку `utf8_unicode_ci` (рис. 7).

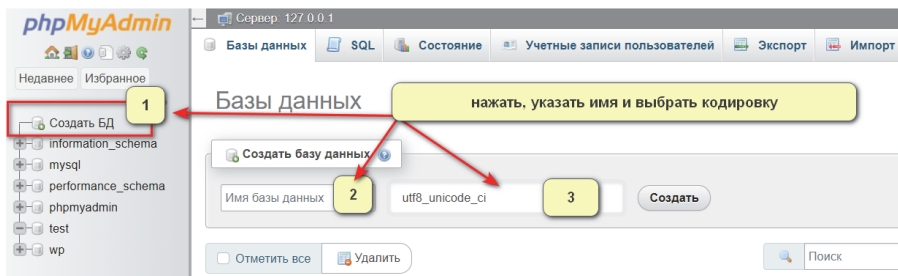


Рис. 7. Создание базы данных

Затем необходимо создать таблицу. Назовём ее `news`, укажем, что нам необходимо 6 полей. В таблице сделаем первичный ключ – поле «`id`», у него нужно отметить «`primary_key`» и «`A_I`» (автоинкремент, т. е. значение поля само увеличивается на 1 при добавлении новости). Остальные поля делаем как на рис. 8.

Имя	Тип	Длина/Значения	По умолчанию	Сравнение	Атрибуты	Null	Индекс
id <small>Выбрать из центральных столбцов</small>	INT		Нет			<input type="checkbox"/>	PRIMARY PRIMARY <input checked="" type="checkbox"/>
d <small>Выбрать из центральных столбцов</small>	TIMESTAMP		CURRENT_TIME			<input type="checkbox"/>	---
name <small>Выбрать из центральных столбцов</small>	TEXT		Нет			<input type="checkbox"/>	---
stext <small>Выбрать из центральных столбцов</small>	TEXT		Нет			<input type="checkbox"/>	---
ftext <small>Выбрать из центральных столбцов</small>	TEXT		Нет			<input type="checkbox"/>	---
img <small>Выбрать из центральных столбцов</small>	TEXT		Нет			<input type="checkbox"/>	---

Рис. 8. Создание таблицы новостей

Далее покажем, как получать и заносить данные в таблицу новостей.

1. Подключение к серверу баз данных

Как правило, подключение к серверу баз данных происходит на каждой странице сайта, поэтому есть смысл вынести данный блок в отдельный файл или отдельную функцию и подключать везде. Для XAMPP команда будет следующей:

```
$mysqli = new mysqli("localhost", "root", "", "mysite");
```

Также стоит включить нужную кодировку при обмене с СУБД:

```
$mysqli->query("set names utf8");
```

2. Получение списка новостей (отсортированные по дате в обратном порядке, первые 10 новостей):

```
$result = $mysqli->query("SELECT * FROM newst
ORDER BY d DESC LIMIT 0,10");
?>
```

Количество полученных новостей: `<?=$result->num_rows?>`

```
<?php
foreach ($result as $row) {
    ?>
```

```

        <p>Новость от <?=$row['d']?>, заголовок:
        <?=$row['name']?></p>
        <?php
    }

```

3. Подготавливаемый запрос для получения конкретной новости + проверка идентификатора на int:

```

    $id=(int) $_GET["id"];
    $stmt = $mysqli->prepare("SELECT * FROM news
WHERE id = ?");
    $stmt->bind_param("i", $id);
    $stmt->execute();
    $result = $stmt->get_result();
    if ($result->num_rows == 1) {
        $row = $result->fetch_assoc();
        ?>
        <p>Новость от <?=$row['d']?>, заголовок:
        <?=$row['name']?></p>
        <?php
    } else ...

```

4. Добавление новой новости:

```

    $stmt = $mysqli->prepare("INSERT INTO news
(d,name,stext,ftext,img) VALUES (?, ?, ?, ?, ?);
    $stmt->bind_param("sssss", $_POST["name"], ....
);
    $stmt->execute();

```

Обратите внимание, мы не указываем при добавлении поле id, так как оно присвоится автоматически. Также мы храним изображение в виде пути к нему на диске, подробнее – в документации <https://www.php.net/manual/ru/features.file-upload.post-method.php>.

5. Другие запросы, которые могут вам понадобиться:

Удалить новость с конкретным id: DELETE FROM NEWS WHERE id=?

Обновить новость с конкретным id: UPDATE news SET name=?, stext=?, WHERE id=?

СОЗДАНИЕ СТРУКТУРЫ ФАЙЛОВ ДЛЯ САЙТА НА PHP

Для создания структуры файлов для сайта существуют два основных подхода:

1. Единый файл контроллера, в котором по запросу выполняется та или иная функция.
2. Отдельные файлы для каждой страницы / действия сайта, которые подключают фрагменты шаблона сайта.

Естественно, что допустимы и комбинации этих вариантов, когда под большую задачу создается отдельный файл, в котором уже выбирается действие под каждую подзадачу.

Покажем на примере отличия этих двух вариантов.

В нашем примере новостной сайт должен выполнять следующие задачи: отображать список новостей, детальную новость, разрешать вход под администратором, добавлять новость, редактировать новость, удалять новость.

Также должно осуществляться подключение к базе данных для работы с новостями.

ВАРИАНТ С ЕДИНЫМ ФАЙЛОМ КОНТРОЛЛЕРА

Для простоты демонстрации объединим контроллер и шаблон сайта. Создадим файл шаблона. Его нужно назвать `index.php`.

Здесь и далее мы используем короткую форму старта PHP – “<?” вместо “<?php”.

Передаваемые параметры могут быть:

`id` – идентификтор новости;

`w` – действие, которое необходимо сделать. По умолчанию мы показываем главную страницу со списком новостей;

и другие параметры, в зависимости от страницы.

<?

```
//подключиться к базе данных, старт сессии, проверка авторизации пользователя  
session_start();  
require("functions.php"); //подключаем свою библиотеку функций
```

```

//подключаемся к MySQL, пользователь root без паро-
ля, база данных site
$mysqli = new mysqli("localhost", "root", "",
"site");
if ($mysqli->connect_errno) {
    echo "Не удалось подключиться к MySQL: (" .
$mysqli->connect_errno . ") " . $mysqli-
>connect_error;
}
// проверка, авторизован ли пользователь под адми-
нистратором ранее
if (isset($_SESSION['admin']) &&
($_SESSION['admin']>0))
    $is_admin=1;
else
    $is_admin=0;
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" con-
tent="text/html; charset=utf-8" />
        <title>Новостной сайт</title>
        <link href="style/site.css" rel="stylesheet">
    </head>
    <body>
        <div id="container">
            <div id="header">Новости</div>
            <div id="sidebar">
                <p><a href="index.php">Список новостей</a></p>
                <p><a href="index.php?w=login">Войти в
систему</a></p>
                <p><a href="index.php?w=about">О сайте</a></p>
            </div>
            <div id="content">

```

```

<?
//проверяем запрошенное действие и вызываем соот-
ветствующую функцию
if (isset($_REQUEST['w']))
    $w=$_REQUEST['w'];
else
    $w="show_news"; //обязательно нужно присваивать
значение по умолчанию
//а теперь вызываем нужную функцию
switch ($w) {
    case "add": //показать форму добавления новости
        if ($is_admin) {
            show_html_for_add_news();
        }
        break;
    case "add2": //сохранить введенные значения
        if ($is_admin) {
            save_news_to_db();
        }
        break;
    ... //другие варианты
    default: //по умолчанию показать список новостей
        show_all_news();
}
?>
</div>
<div id="footer">&copy; НГТУ</div>
</div>
</body>

```

Обратите внимание, что в приведенном примере неочевидно, как можно установить заголовок окна браузера (тэг title) в соответствии с выбранной задачей.

Также современные требования к SEO предполагают, что у каждой значимой страницы будет свой «физический адрес».

Обе задачи уже неоднократно решены в современных CMS системах. Например, для заголовка страницы в Bitrix Framework работает механизм «отложенных функций», когда контент выводится в шаблон

не сразу, а в конце выполнения программы. Также популярным методом является разделение программной части и визуального отображения, когда сначала подготавливаются все данные, а затем заполняется шаблон страницы.

Для формирования же «физического адреса» используется механизм ЧПУ (или SEF) адресов, когда несуществующие адреса страниц перехватываются программным путем и параметры адреса используются для создания контента. Например, новостной сайт [lenta.ru](https://lenta.ru/news/2024/08/21/pod-novosibirskom-nashli-drevnee-more) имеет такие адреса страниц:

<https://lenta.ru/news/2024/08/21/pod-novosibirskom-nashli-drevnee-more>.

Здесь:

- lenta.ru – адрес сайта;
- [news](https://lenta.ru/news/) – раздел сайта;
- [2024/08/21](https://lenta.ru/news/2024/08/21/) – дата новости, можно открыть этот адрес и увидеть все новости дня;
- [pod-novosibirskom-nashli-drevnee-more](https://lenta.ru/news/2024/08/21/pod-novosibirskom-nashli-drevnee-more) – «символьный код» новости, для отображения конкретной новости.

На большинстве подобных сайтов данных папок в реальности не существует. Система программным образом перехватывает запрошенный адрес и по нему делает запрос к базе данных для отображения конкретной новости или списка новостей.

ВАРИАНТ С РАЗНЫМИ СТРАНИЦАМИ ДЛЯ РАЗНЫХ ДЕЙСТВИЙ ПОЛЬЗОВАТЕЛЯ

Рассмотрим вариант, когда под каждое действие будет своя страница.

Как правило, в этом случае шаблон делится на файлы `header` и `footer` (или большее количество файлов) и они подключаются из страницы:

Файл `header.php`

```
<?
```

```
//подключиться к базе данных, старт сессии, проверка авторизации пользователя
```

```
session_start();
```

```
require("functions.php"); //подключаем свою библиотеку функций
```

```
//подключаемся к MySQL, пользователь root без пароля, база данных site
```



```

$mysqli = new mysqli("localhost", "root", "",
"site");
if ($mysqli->connect_errno) {
    echo "Не удалось подключиться к MySQL: (" .
$mysqli->connect_errno . ") " . $mysqli-
>connect_error;
}
// проверка, авторизован ли пользователь под адми-
нистратором ранее
if (isset($_SESSION['admin']) &&
($_SESSION['admin']>0))
    $is_admin=1;
else
    $is_admin=0;
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" con-
tent="text/html; charset=utf-8" />
        <title><?=$title?></title>
        <link href="style/site.css" rel="stylesheet">
    </head>
    <body>
        <div id="container">
            <div id="header">Новости</div>
            <div id="sidebar">
                <p><a href="index.php">Список новостей</a></p>
                <p><a href="login.php">Войти в систему</a></p>
                <p><a href="about.php">О сайте</a></p>
            </div>
            <div id="content">

Файл footer.php:
        </div>

```

```
<div id="footer">&copy; НГТУ</div>
</div>
</body>
```

Файл страницы сайта (например, index.php):

```
<?
$title="Новостной сайт";
require("header.php");

//вывести список новостей
...

require("footer.php");
?>
```

Файл другой страницы сайта (например, add.php):

```
<?
$title="Добавление новости";
require("header.php");

//вывести HTML для добавления новости
...

require("footer.php");
?>
```

Естественно, что представленные варианты можно комбинировать, использовать шаблонизаторы, CMS, framework и другие средства, упрощающие и упорядочивающие разработку.

Задание на самостоятельную работу: выберите подходящий для вас способ реализации и запрограммируйте новостной сайт.

ИСПОЛЬЗОВАНИЕ JavaScript и jQuery

Как мы указывали раньше, для реализации динамических сценариев в frontend части используется язык JavaScript.

Стоит отметить, что JavaScript не имеет ничего общего (кроме некоторых элементов синтаксиса) с Java и нельзя их путать между собой.

Структурно JavaScript можно представить в виде объединения трех четко отделяемых друг от друга частей:

- ядро (стандарт ECMAScript);
- объектная модель браузера (Browser Object Model или BOM (англ.));
- объектная модель документа (Document Object Model или DOM).

Если рассматривать JavaScript в отличных от браузера окружениях (например, серверный JavaScript), то объектная модель браузера и объектная модель документа могут не поддерживаться.

Таким образом, программы на языке JavaScript, помимо собственно выполнения программного кода, имеют доступ к объектам браузера и к объектам HTML документа. На JavaScript можно, например, изменить, добавить, удалить какой-либо объект HTML документа. Именно таким образом реализуются интерактивные веб-страницы и элементы на них: слайдеры, меню, онлайн добавление или удаление блоков на странице и т. д.

Рассмотрим встраивание JavaScript в веб-страницы.

Для добавления JavaScript-кода на страницу можно использовать теги `<script></script>`, которые рекомендуется, но не обязательно, помещать внутри контейнера `<head>`. Контейнеров `<script>` в одном документе может быть сколько угодно. Атрибут `<type='text/javascript'>` указывать не обязательно, данное значение используется по умолчанию.

Пример: Скрипт, выводящий модальное окно с классической надписью «Hello, World!» внутри браузера:

```
<script type="application/javascript">
  alert('Hello, World!');
</script>
```

Расположение внутри тега

Спецификация HTML описывает набор атрибутов, используемых для задания обработчиков событий. Пример использования:

```
<a href="delete.php" onclick="return confirm('Вы уверены?'); return false;">
  Удалить
</a>
```

В приведенном примере при нажатии на ссылку функция `confirm('Вы уверены?')` вызывает модальное окно с надписью «Вы уверены?», а `return false;` блокирует переход по ссылке. Разумеется,

этот код будет работать только, если в браузере есть и включена поддержка JavaScript, иначе переход по ссылке произойдет без предупреждения.

Использование кода JavaScript в контексте разметки страницы расценивается в рамках ненавязчивого JavaScript как плохая практика. Аналогом приведенного примера может являться, например, следующий фрагмент JavaScript. Для этого необходимо у ссылки указать идентификатор alertLink и реализовать обработчик по нажатию на объект с этим идентификатором:

HTML код:

```
<a href="delete.php" id="alertLink">
    Удалить
</a>
```

JavaScript:

```
window.onload = function() {
    var linkWithAlert = document.getElementById("alertLink");
    linkWithAlert.onclick = function() {
        return confirm('Вы уверены?');
    };
};
```

Поэтому чаще всего JavaScript выносят в отдельный файл.

Есть и третья возможность подключения JavaScript – написать скрипт в отдельном файле, а потом подключить его с помощью конструкции

```
<head>
    <script type="application/javascript"
src="http://Путь_к_файлу_со_скриптом">
    </script>
</head>
```

При этом изначально JavaScript задумывался как язык, на котором будет достаточно просто программировать не-программистам, дизайнерам, верстальщикам и т. д.

Частично это так и произошло, однако не-программисты требовали еще большего упрощения. И для еще большего упрощения были придуманы JS-фреймворки.

Например, один из самых популярных JS-преймворков – jQuery (www.jquery.com) позволяет, воспользовавшись несложной инструкцией, получить на своем сайте набор интерактивных элементов. Это стало возможным благодаря большому числу плагинов, в том числе бесплатных.

Пример работы с jQuery:

Работа с элементами DOM:

Получить элемент `<button>` с классом `'continue'` и изменить на нем надпись на `'Next Step...'`:

```
$( "button.continue" ).html( "Next Step..." )
```

Обработка событий:

Показать элемент, скрытый свойством CSS `display:none` с `id=#banner-message` при нажатии любой кнопки в элементе HTML с `id=#button-container`:

```
var hiddenBox = $( "#banner-message" );
$( "#button-container button" ).on( "click", function( event ) {
    hiddenBox.show();
});
```

Ajax

Вызвать скрипт на сервере с адресом `/api/getWeather`, передать ему параметр GET запроса `zipcode=97201` и заменить элемент HTML с `id=#weather-temp` на возвращенный AJAX запросом текст:

```
$.ajax({
    url: "/api/getWeather",
    data: {
        zipcode: 97201
    },
    success: function( result ) {
        $( "#weather-temp" ).html( "<strong>" + result + "</strong> degrees" );
    }
});
```

Подключение плагина

Подключим к нашей странице плагин `fancybox` – отображение всплывающей картинки при нажатии на маленькое изображение на странице (<http://fancyapps.com/fancybox/3>).

Для начала, разберемся, как это работает.

1. Вам потребуется две картинки: маленькая – для размещения на странице и большая – для отображения во всплывающем окне. Важно, чтобы на странице изначально не было очень больших по размеру (в килобайтах) картинок, так как это замедляет загрузку картинок.

2. На странице каждую картинку, для которой должно быть всплывание, оборачиваем в тег `<a>` и у тега `<a>` указываем атрибут `data-fancybox`. В этом случае плагин автоматически применится к указанным картинкам.

3. В случае, если JavaScript отключен или плагины не смогли загрузиться, картинка откроется в новом окне.

Итог в коде HTML будет следующим:

```
<!-- 1. Подключить последние версии jQuery и fancybox -->
<script src="//code.jquery.com/jquery-3.3.1.min.js"></script>
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/gh/fancyapps/fancybox@3.5.7/dist/jquery.fanc
ybox.min.css" />
<script
src="https://cdn.jsdelivr.net/gh/fancyapps/fancybox@3.5.7/dist/jquery.fanc
ybox.min.js"></script>
<!-- 2. В коде HTML создать ссылки -->
<a data-fancybox="gallery" href="big_1.jpg"></a>
<a data-fancybox="gallery" href="big_2.jpg"></a>
```

РАЗДЕЛЕНИЕ КОДА И ПРЕДСТАВЛЕНИЯ, ШАБЛОНИЗАТОРЫ

Наверное, вы заметили, что и в нашем учебном пособии, и в документации к языкам программирования, и в обучающих курсах часто встречаются рекомендации по оптимальному встраиванию кода используемых языков в HTML и генерации HTML кода программным путем. Одна из причин этого в том, что и PHP, и JavaScript изначально создавались для использования непрофессиональными программистами. Последствием широкой популярности явилось то, что многие PHP и JavaScript программы представляют смесь HTML, PHP и JavaScript, в которой очень сложно разобраться, а результат сложно

поддерживать. Для исключения этой ситуации в веб-разработке часто используется модель MVC.

Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») – схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо.

- Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя свое состояние.
- Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
- Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Основная цель применения этой концепции состоит в отделении бизнес-логики (модели) от ее визуализации (представления, вида). За счет такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи.

1. К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы.

2. Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью по кнопке, ввод данных) – для этого достаточно использовать другой контроллер.

3. Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

Для реализации такой модели часто применяют шаблонизаторы. Их задача – исключить из визуального представления (View) возможность реализовывать бизнес-логику.

К шаблонизаторам PHP относят сам PHP, Smarty, Blade и др.

Даже если вы не используете шаблонизатор, а реализуете всю логику в PHP, вы можете существенно повысить читаемость вашего кода, применяя простые правила.

1. Всю подготовку данных и работу с бизнес-логикой имеет смысл вынести в функции либо в отдельный файл, либо в начало PHP скрипта.

2. Если вам нужно вывести HTML код, желательно закрыть PHP блок (директива “?”), вывести HTML, а затем возобновить PHP. Альтернативный, нерекомендуемый способ – вывод HTML с помощью оператора echo – существенно усложняет чтение текста и его модификацию.

3. Если вам нужно вывести переменную внутри HTML блока, рекомендуем использовать короткую форму оператора echo: `<?=$var_name?>`

Это позволяет очень наглядно форматировать документ.

Аналогичные советы применимы для HTML, JavaScript и CSS.

1. Все стили следует задавать в CSS.

2. Везде, где это оправданно, JavaScript следует размещать в отдельном файле, вызывая его через функции – обработчики событий.

ИСПОЛЬЗОВАНИЕ CMS

Следующим большим шагом развития веб-разработки стало появление Content Management System – систем управления контентом, первая задача которых была стандартизировать и существенно снизить затраты на разработку веб-сайтов за счет использования стандартных блоков, внутренних структур данных и модулей.

Например, в 1С-Битрикс при создании инфоблока (аналог таблицы в базе данных) вы сразу получаете набор полей, наиболее применимых в веб-разработке:

- Id.
- Название.
- Дата начала активности и дата завершения активности (применяется для новостей и статей).
- Картинка для анонса и детальная картинка.
- Текст для анонса и детальный текст.
- И т. д.

Такие объекты данных можно создавать без знания программирования и подключать к ним компоненты, отображающие эти данные в виде статей, новостей, товаров и т. д.

Также в CMS реализуется создание структуры сайта, управление пользователями и т.д.

Все популярные CMS имеют возможность расширения за счет подключаемых модулей, а библиотека доступных сторонних модулей для них насчитывает сотни и тысячи штук.

Популярные современные CMS:

1С-Битрикс, Wordpress, Joomla, Drupal и др.

ИСПОЛЬЗОВАНИЕ PHP FRAMEWORK

Возможны ситуации, когда сборка сайта из готовых блоков и настройка их будет недостаточна для решения задачи. Либо использование CMS является избыточным для решения специфических задач веб-приложения (как правило, это уже не сайт). В этом случае разработка осуществляется либо напрямую на выбранном языке программирования, либо с использованием фреймворка. Справедливости ради отметим, что большинство современных CMS систем содержат в себе собственный или внешний фреймворк для разработки необходимых дополнительных модулей.

Выделим основные случаи, когда использование фреймворка предпочтительнее использования CMS системы.

1. Когда разрабатывается не сайт, а веб-система, веб-сервис или другая сложная система, в которой мало применимы классические объекты сайта (страницы, новости, товары и т.д.)

2. Когда веб-приложение должно работать очень быстро, и генерируемые им данные не являются страницами в полном смысле этого слова (т. е. не содержат дизайн и прочее). Пример – мессенджеры, сервисы обмена данными.

В этом случае имеет смысл выбрать язык программирования и фреймворк, которые наиболее подходят для решения вашей задачи.

РАЗМЕЩЕНИЕ ПРОЕКТА НА ХОСТИНГЕ

В предыдущих работах мы использовали локальный сервер – то есть и файлы, и веб-сервер были фактически на одном компьютере.

Как правило, сайты хранятся на удалённом сервере с доступом по ssh/sftp. Существует много плагинов для редакторов кода, которые позволяют работать с файлами в этом режиме. Вы можете подключить их самостоятельно.

РАЗРАБОТКА СОБСТВЕННОГО ВЕБ-ПРИЛОЖЕНИЯ

Для того чтобы освоить создание собственного веб-приложения (или модуля к сайту), предлагаем реализовать одно из следующих заданий.

1. Фотоконкурс

Фотоконкурс – это интерактивный раздел сайта, в котором зарегистрированные пользователи сайта могут размещать свою заявку на участие (может состоять из нескольких фото), а другие посетители могут голосовать посредством «лайков» в соцсетях.

У каждого конкурса есть период конкурса, в который можно подавать заявки. После этого можно только смотреть результаты (также можно голосовать, но голоса уже не учитываются).

Реализовать возможности для администратора сайта:

- а) создавать новые фотоконкурсы (например, «Лучшее новогоднее фото») с указанием описания и срока, до которого проводится конкурс;
- б) одобрять или снимать с публикации заявки;
- в) смотреть результаты и выбирать победителя;
- г) указывать описание конкурса и итоговый отчет о вручении призов.

Реализовать возможности для участника конкурса:

- а) зарегистрироваться на сайте (желательно с входом через социальные сети);
- б) подать заявку на участие в конкурсе. К заявке можно прикрепить несколько фотографий и текст. Желательно реализовать возможность кадрировать основную фотографию для красивого отображения в списке участников.

Реализовать возможности для посетителя:

- а) посмотреть всех участников – по рейтингу или по дате добавления;

б) проголосовать за участника через свой аккаунт в социальных сетях.

2. Реализовать акцию, позволяющую накапливать баллы и выбирать доступные призы.

Акция – это интерактивный раздел сайта, в котором зарегистрированные пользователи сайта могут регистрировать найденные на упаковке товаров коды, накапливать баллы и получать за них призы из каталога.

У каждой акции есть период, в который можно накапливать баллы и делать заявку на призы. После этого можно только смотреть результаты.

Реализовать возможности для администратора сайта:

а) создавать новую акцию (например, «Любимые молочные продукты» – марка) с указанием описания и срока, до которого проводится акция;

б) смотреть результаты по накоплению баллов и заказанным призам;

в) указывать описание акции и итоговый отчет о вручении призов;

г) создавать и наполнять каталог призов (у каждого приза есть фото, описание и стоимость – количество баллов);

д) указывать шаблон промо-кода (или конкретный список промо-кодов) с указанием баллов за каждый код.

Реализовать возможности для участника конкурса:

а) зарегистрироваться на сайте (желательно с входом через социальные сети);

б) регистрировать промо-коды и получать за них баллы;

в) заказывать товары из каталога за баллы.

3. Реализовать внутреннюю переписку между зарегистрированными пользователями сайта.

Аналог – сообщения «ВКонтакте».

Реализовать возможности:

а) выбирать пользователя, с которым необходимо общаться;

б) отправлять текстовые сообщения, картинки, видео и документы;

в) смотреть список всех диалогов;

г) смотреть историю сообщений.

ПРИЛОЖЕНИЕ

Использование панели разработчика в браузерах

У многих современных браузеров присутствует полезный для веб-разработчика функционал, называемый, как правило, панелью инструментов разработчика (или каким-либо похожим образом). Иногда этот функционал реализован в виде соответствующих плагинов (plug-in) для данного браузера.

В таких популярных браузерах, как Mozilla Firefox и Google Chrome, инструментарий разработчика можно вызвать с помощью клавиши **F12** (рис. 9 и 10).

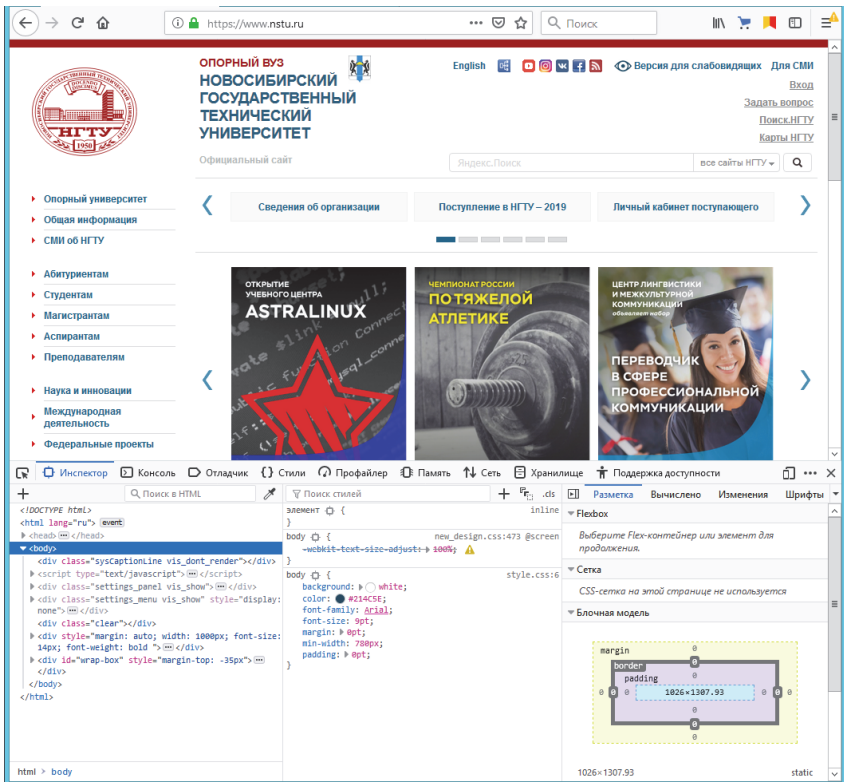


Рис. 9. Панель разработчика в Mozilla Firefox

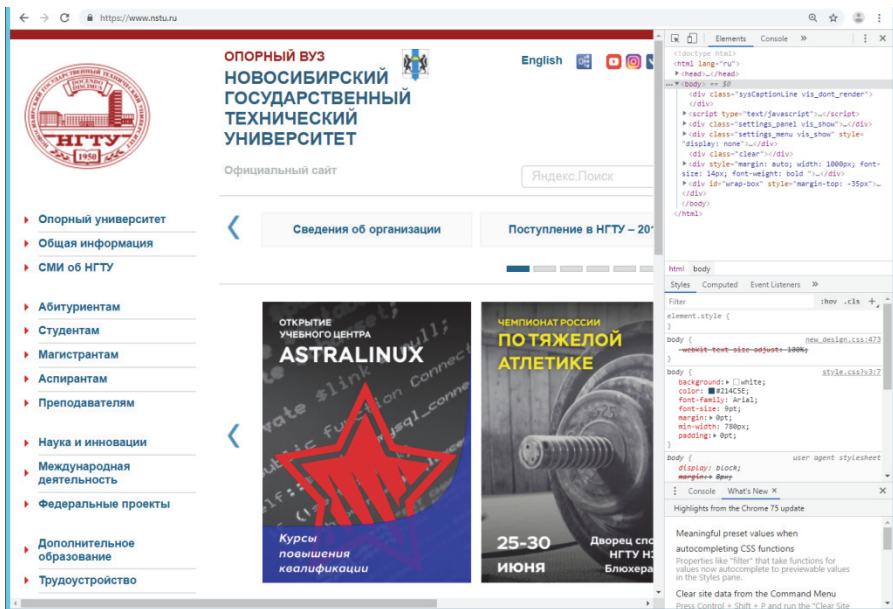




Рис. 10. Панель разработчика в Google Chrome

Рассмотрим для примера панель разработчика Google Chrome. Первый элемент панели  позволяет с помощью указателя мыши проанализировать элементы, расположенные на странице. При этом, если включен элемент панели «Elements» (как на рис. 10), то ниже будет выделен соответствующий выделенному элементу код. Вообще элемент панели «Elements» отображает html-структуру документа и все, что в нее включено (CSS, JavaScript). Если кликнуть левой кнопкой мыши на строчку кода, то впереди нее появится троеточие, по которому также можно кликнуть левой кнопкой мыши, после чего появится контекстное меню (рис. 11), позволяющее отредактировать соответствующий код и увидеть результат этих изменений в основном окне браузера.

Второй элемент панели  позволяет увидеть, как будет выглядеть страница на устройствах с другим разрешением экрана. При этом дополнительно появляется панель устройств (DevTools, рис. 12), которая позволяет задать точные свойства экрана устройства. Справа

на панели устройств есть вертикальное троеточие, по которому также можно кликнуть левой кнопкой мыши для дополнительных действий.

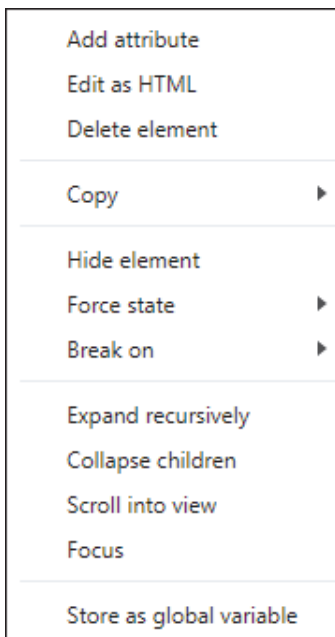


Рис. 11. Контекстное меню изменения кода



Рис. 12. Панель DevTools

Элемент панели «Console» (рис. 13) предназначен для отображения сообщений (предупреждений, ошибок), выводимых разными динамическими элементами страницы. Также в консоли можно выполнять код JavaScript. При выборе этого элемента дополнительно появляется панель, позволяющая настроить параметры работы с данным элементом.

Элемент «Sources» (рис. 14) позволяет увидеть все загруженные со страницей картинки и скрипты и получить доступ к этим файлам.

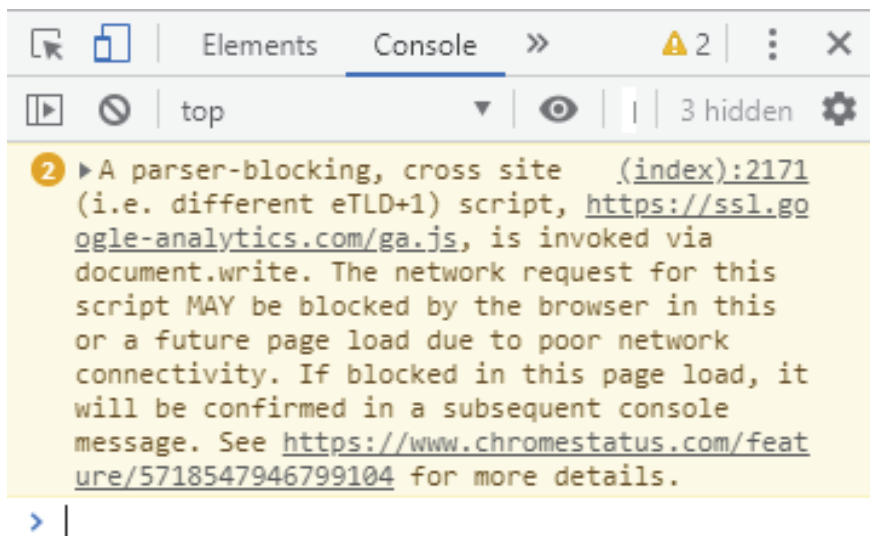


Рис. 13. Элемент «Console»

Элемент «Network» (рис. 15) является очень мощным инструментом для анализа обмена данными между браузером и веб-сервером. Здесь вы можете увидеть, какими данными обменивались браузер и веб-сервер и как во времени загружались различные элементы страницы, и понять, какие элементы тормозят загрузку страницы и нуждаются в переработке.

Другие элементы панели

Элемент «Performance» (рис. 16) отображает использования сети, выполнения JavaScript кода и загрузки памяти во времени.

Элемент «Memory» (рис. 17) предназначен для отслеживания нагрузки, которую оказывает выполнение кода на систему.

Элемент «Application» (рис. 18) используется для инспектирования и очистки всех загруженных ресурсов.

Элемент «Security» позволяет посмотреть протокол безопасности и данные о сертификате безопасности, если они имеются.

Элемент «Audits» анализирует, как загружается страница, и затем предоставляет предложения по оптимизации.

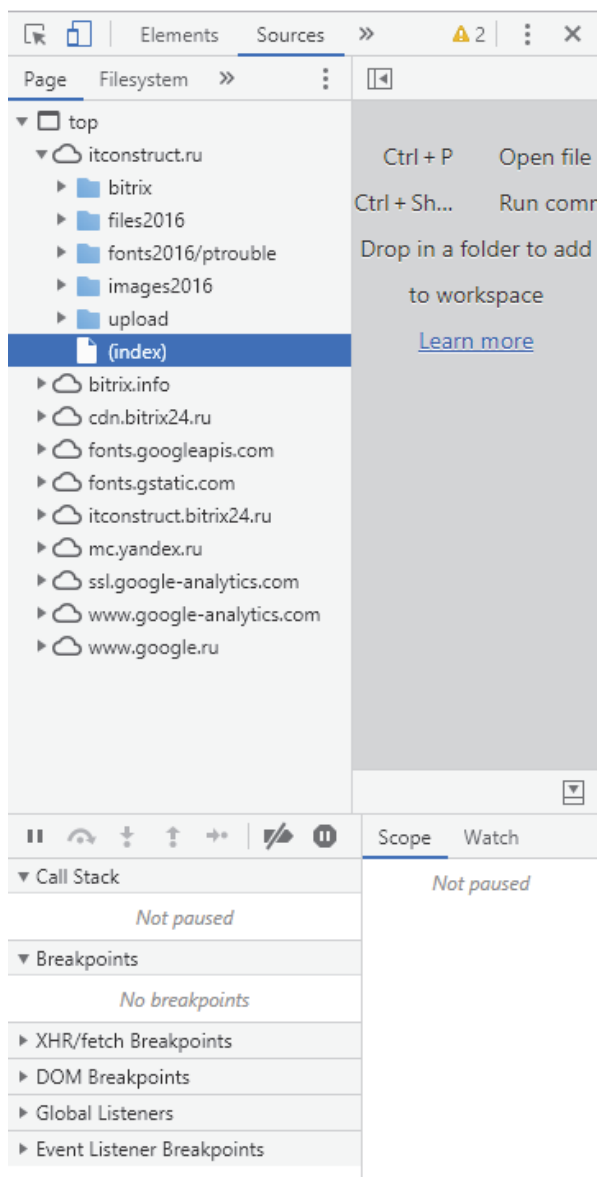


Рис. 14. Элемент «Sources»

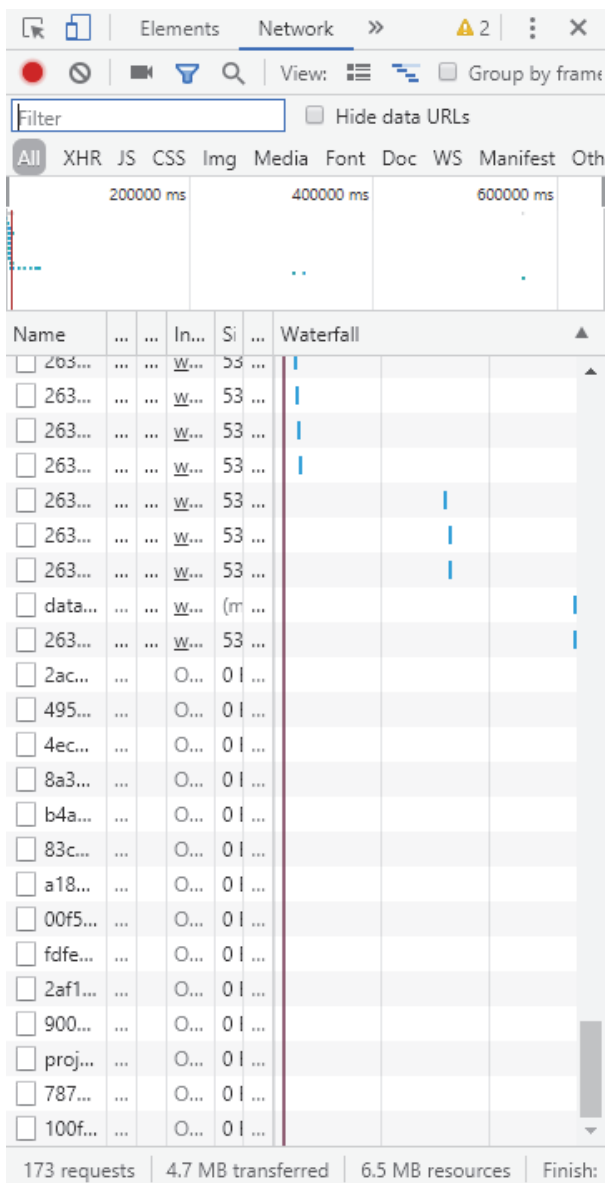


Рис. 15. Элемент «Network»

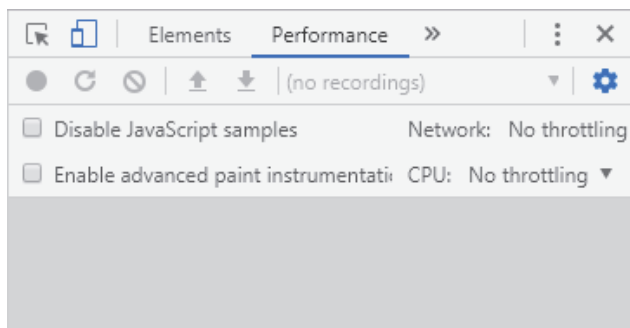


Рис. 16. Элемент «Performance»

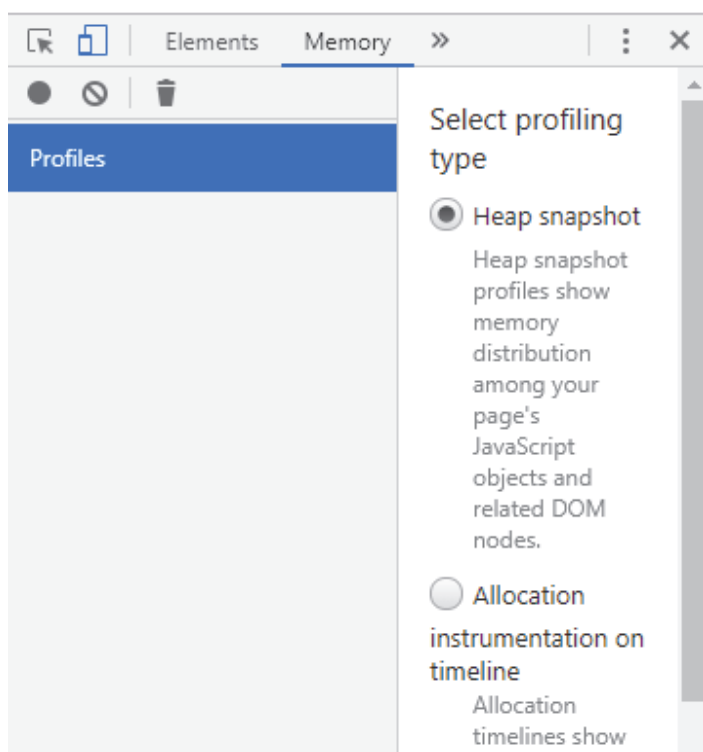


Рис. 17. Элемент «Memory»

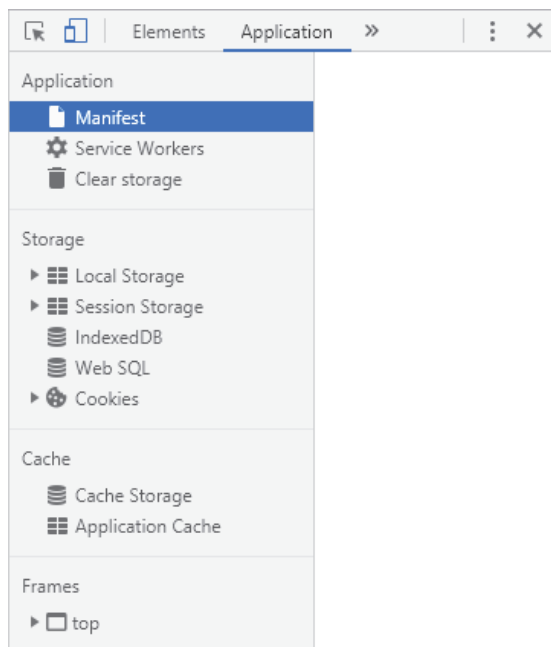


Рис. 18. Элемент «Application»

ЗАКЛЮЧЕНИЕ

В заключение отметим, что в отличие от таких классических дисциплин, как, например, «Математический анализ», теоретический и практический материал для дисциплины «Современные технологии разработки веб-приложений» постоянно обновляется (за исключением самих базовых принципов) вслед за активно продолжающимся развитием веб-технологий и требует постоянной актуализации. В связи с этим изучающему данный курс рекомендуется постоянно обращаться к авторитетным источникам (ссылки на них встречались в пособии), как говорится, «для сверки часов».

Авторы надеются, что данное учебное пособие поможет его читателям проложить себе путь в увлекательный и бурно развивающийся мир современных веб-технологий, в котором востребованность специалистов год от года только увеличивается.

ОГЛАВЛЕНИЕ

Введение	3
О веб-разработке	3
Современное состояние в области веб-разработки.....	3
Основная цель курса	4
Типы веб-проектов.....	5
Архитектура веб-приложения.....	7
О структуре подачи материала в данном учебном пособии	8
Языки разработки веб-приложений	9
Frontend	9
Backend.....	10
Специфика работы веб-приложений.....	11
Упрощенная схема работы веб-приложения	11
Запуск и жизненный цикл серверной веб-программы	13
Хранение и передача данных во время визита пользователя.....	13
SPA и MPA приложения.....	15
Скорость выполнения	16
Занимаемая память.....	17
Безопасность	18

Взаимодействие с СУБД.....	20
Кэширование	20
Этапы разработки веб-сайта	21
Типовой процесс разработки сайта (классический).....	21
Техническое задание.....	22
Разработка прототипов страниц.....	23
Дизайн	24
HTML верстка.....	26
Подключение frontend скриптов	26
Разработка backend части	26
Тестирование и запуск проекта.....	27
Развитие веб-проекта	27
Практические задания	27
Общее Задание на проект	28
Последовательность действий для выполнения практических работ	29
Установка веб-сервера.....	29
Создание страниц на HTML и CSS.....	30
Подключение CSS Framework для придания новостному сайту современного вида	31
Создание новостного сайта на HTML, CSS, PHP, MySQL	32
Основы PHP	32
Хороший и плохой стиль сочетания PHP И HTML	33
Запросы к базам данных и ассоциативные массивы	34
Создание базы данных и таблиц для новостного сайта	34

Создание структуры файлов для сайта на PHP	37
Вариант с единым файлом контроллера.....	37
Вариант с разными страницами для разных действий пользователя	40
Использование JavaScript и jQuery	42
Разделение кода и представления, шаблонизаторы	46
Использование CMS	48
Использование PHP Framework	49
Размещение проекта на хостинге.....	49
Разработка собственного веб-приложения	50
Приложение.....	52
Заключение.....	60
Оглавление	61

**Петров Роман Владимирович
Вагин Денис Владимирович**

**СОЗДАНИЕ СОВРЕМЕННЫХ КРОССПЛАТФОРМЕННЫХ ПРИЛОЖЕНИЙ
НА ОСНОВЕ WEB-ТЕХНОЛОГИЙ**

Учебное пособие

В авторской редакции

Выпускающий редактор *И.П. Брованова*
Дизайн обложки *А.В. Ладыжская*
Компьютерная верстка *Н.В. Гаврилова*

Налоговая льгота – Общероссийский классификатор продукции
Издание соответствует коду 95 3000 ОК 005-93 (ОКП)

Подписано в печать 24.09.2024. Формат 60 × 84 1/16. Бумага офсетная. Тираж 50 экз.
Уч.-изд. л. 3,72. Печ. л. 4,0. Изд. № 119. Заказ № 177. Цена договорная

Отпечатано в типографии
Новосибирского государственного технического университета
630073, г. Новосибирск, пр. К. Маркса, 20