Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Алтайский государственный технический университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Крайванова В.А., Третьяков А.А.

Учебно-методическое пособие «Современные средства разработки ПО»

для студентов очной формы обучения
направления 09.04.03 Программная инженерия

Часть I

Барнаул 2019

УДК 004.42

Крайванова В.А., Третьяков А.А. Учебно-методическое пособие «Современные средства разработки ПО» / В.А. Крайванова, А.А. Третьяков; АлтГТУ им. И.И. Ползунова. – Барнаул, АлтГТУ, 2019. – 31 с.

Учебно-методическое пособие содержит теоретический материал и методические указания к лабораторным работам по дисциплине «Современные средства промышленной разработки ПО» и предназначено для студентов, обучающихся по направлению 09.04.03 «Программная инженерия».

Содержание

Введение

Разработка программного обеспечения - огромная и очень быстро меняющаяся индустрия, требующая огромное количество человеческих ресурсов. Еще недавно любой выпускник университета с профессией программиста легко находил себе высокооплачиваемую работой даже без опыта. Сегодня ситуация несколько изменилась. На рынке труда одновременно сосуществует большое количество кандидатов, ищущих работу, и большое количество незакрытых вакансий. Одна из причин этого кроется в неуниверсальности знаний и навыков, имеющихся у кандидатов. Современный программист должен сочетать в себе как знание конкретных инструментов, так и умение быстро осваивать новые.

Задайте себе несколько вопросов. Сколько языков программирования вы знаете и на скольких из них вы готовы начать писать уже завтра? Сегодня пройти собеседование на вакансию junior-программиста, а завтра прийти в компанию, сесть и начать писать код. Как вы думаете, сколько стоит ваш язык программирования? Как много вам нужно узнать чтобы перестать быть junior и стать middle-программистом или senior-программистом? Примут ли вас на должность junior или стажера, и что делать, если в вашем городе или поблизости нет вакансии с вашим языком программирования? Есть ли вакансии на удаленную работу?

А теперь представьте эту проблему с другой стороны: вы видите вакансию с потрясающими условиями, требуются только базовые знания какой-то технологии, например, языка Erlang или Kotlin. Вы готовы изучить Erlang ради хороших условий труда? Насколько ценным специалистом вы собираетесь стать?

Существует очень много таких вопросов и чтобы давать на них ответы, которые будут вас радовать, необходимо постоянно расширять свой кругозор и осваивать новые технологии. Именно этому посвящен данный курс.

Данный курс преследует три цели, две из них связаны с расширением теоретических знаний, и одна - с наработкой новых навыков.

1. Получить представление о широком спектре существующих технологий и инструментов разработки ПО. Научиться определять инструмент для решения конкретной задачи. Научиться оценивать качество инструмента и целесообразность вложения времени в освоение нового инструмента.
2. Освоить самые распространенные подходы к построению архитектуры ПО, которые позволяют существенно повысить скорость разработки.
3. Научиться быстро осваивать новые инструменты и языки программирования.

Обратите внимание, что мы не ставим цели углубляться в конкретные технологии разработки, основная задача - познакомиться с многообразием существующих инструментов и научиться х осваивать.

Комбинация широкого кругозора, глубоких фундаментальных знаний в области принципов проектирования ПО и умения быстро осваивать необходимые инструменты - отличный набор для старта в профессии.

1 Основные понятия веб-разработки

Разработка веб-приложения - одна из наиболее стандартных и распространенных задач в программировании.

Огромная индустрия занимается тем, что разрабатывает веб-сайты: от маленьких блогов и интернет-магазинов до огромных порталов, таких как Avito или Yandex, и супер-гигантов Amazon и Google. Однако, если вы не будете заниматься непосредственно разработкой для сети Интернет, скорее всего, вам так или иначе придется иметь дело с этими технологиями. Если вы разрабатываете банковские или Enterprise-системы для бизнеса, то рано или поздно вам придется интегрироваться с корпоративным сайтом. Внутренние ресурсы компании также могут реализовывать интерфейс через браузер, так как это намного дешевле, чем устанавливать на каждый компьютер отдельное приложение. Если вы - мобильный разработчик или разработчик компьютерных игр, вам, скорее всего, потребуется реализовывать взаимодействие с сервером вашего приложения через некоторый API-интерфейс на основе протоколов HTTP И HTTPS. Кроме того, пользовательские интерфейсы для мобильных приложений часто разрабатываются на Java Script. Если вы системный администратор или специалист по анализу данных, то высока вероятность, что вам придется разрабатывать панель мониторинга с веб-интерфейсом.

Широкая распространенность задачи разработки веб-приложений привела к тому, что, во-первых, для нее уже создано и продолжает создаваться гигантское разнообразие инструментов, а во-вторых, решения этой задачи основывается на хорошо обкатанных, универсальных принципах, применимых во всех остальных сферах разработки ПО. К тому же, под задачу разработки веб-интерфейса есть инструменты на всех популярных языках разработки, что делает ее прекрасным тренажером для освоения новых языков программирования.

Вы, скорее всего, уже знакомы с большей частью обсуждаемых в этой главе понятий, такими как протокол HTTP и верстка.

В этой главе мы рассмотрим три прикладных момента.

1. Как устроены веб-приложения и Интернет в целом с точки зрения прикладного программиста. Мы кратко рассмотрим протокол http и его особенности.
2. Как устроен интерфейс между вашим приложением и человеком в Интернете. Мы рассмотрим основы верстки, понятие адаптивной верстки, и ряд инструментов, которые позволяют программистам создавать прототипы интерфейсов без погружения в нюансы html, css и javascript.
3. Как устроена СУБД MySQL. Этот вопрос отличается от двух предыдущих, так как завязан на конкретный инструмент. Большинство из вас уже знакомы с реляционными базами данных, такими как Oracle или PostgreSQL. В работе с такими базами необходимо понимать, во-первых, универсальность лежащих в основе принципов, и во-вторых, специфику отдельных решений. MySQL - одна из наиболее популярных СУБД для веб-разработки.

1.1 Протокол HTTP

Интернет основан на клиент-серверной архитектуре. Как правило, в качестве клиента выступает браузер, который инициализирует взаимодействие и отправляет на сервер запрос для получения или изменения данных. Запрос по сети попадает на сервер, который решает, какое из имеющихся на нем приложений должно обработать запрос, и передает полученный пакет этому приложению. Приложение обрабатывает запрос в соответствии со своей логикой, и формирует ответ сервера, который отправляется по сети к клиенту. Упрощенная схема данного алгоритма представлена на рисунке 1.1.

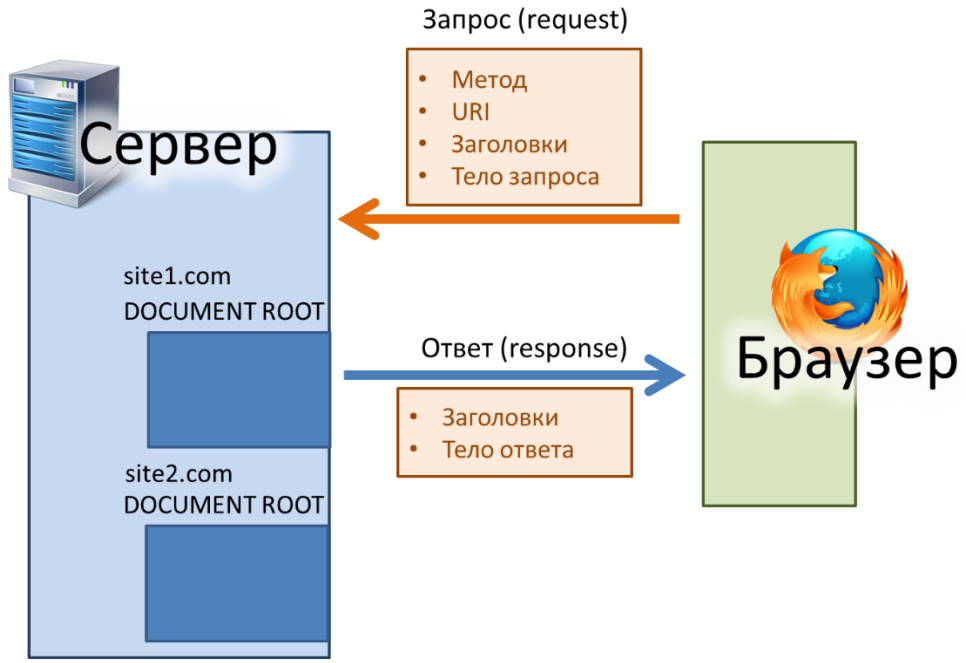


Рисунок 1.1 Упрощенная схема взаимодействия клиента и сервера по протоколу HTTP

Всемирная паутина в основном использует протокол прикладного уровня HTTP (hypertext transport protocol, протокол передачи гипертекста). Этот протокол предполагает передачу информации в виде набора символов, то есть текста. Сетевой пакет HTTP состоит из стартовой строки, набора заголовков и тела запроса.

Стартовая строка запроса клиента содержит:

- метод,
- адрес ресурса в виде URI (Uniform Resource Identifier),
- версию протокола.

Стартовая строка ответа сервера содержит:

- версию протокола,
- 3 цифры - код состояния,
- пояснение (необязательное поле).

Методы и коды состояния очень важны. Фактически, метод запроса является одной из частей адреса, по нему в том числе определяется, какая именно функция на сервере будет обрабатывать этот запрос. Код состояния - это способ, которым сервер оповещает клиента о результате операции: прошла ли она успешно, или случилась ошибка. Ниже мы рассмотрим эти элементы подробнее.

URI может содержать как полный адрес запрашиваемого ресурса, так и адрес внутри сервера. В URI могут быть указаны параметры запроса. В этом случае они идут через знак & в виде пар ключ=значение после знака ? (см. листинг 1.1). Такие URI чаще всего используются на страницах поиска. Для остальных страниц сайта, например, карточек товаров, используются URI, где параметры закодированы в пути на сайте через знак /. Эти два способа кодирования параметров в URI могут использоваться совместно, как в третьем адресе с листинга 1.1.

Листинг 1.1 Примеры URI с параметрами

```
https://www.dns-shop.ru/search/?q=%D0%BD%D0%BE%D1%83%D1%82%D0%B1%D1%83%D0%BA

http://avtomobilistam.ru/vehicles_search.php?what=car&man=44&model=&type=4&gearbox=3&year1=0&year2=2010

https://www.avito.ru/barnaul/kvartiry/prodam/studii?district=167&s_trg=3&user=1

https://www.google.com/search?q=owl&tbs=ic:specific,isc:orange,sur:fc&tbm=isch&tbas=0&source=lnt&sa=X&ved=0ahUKEwihkNXm35vfAhUMw4sKHaSnDG0QpwUIIA&biw=1073&bih=915&dpr=1
```

Для поддержания виртуального хостинга, то есть возможности размещать на одном порту одной машины несколько доменных имен, дополнительно используется заголовок Host, в котором указывается доменное имя сайта.

Заголовок HTTP - это пара ключ-значения, разделенная двоеточием. Каждый заголовок заканчивается переводом строки. Кроме Host в сообщении могут присутствовать и другие заголовки. В запросе клиента могут быть указаны тип версия пользовательской программы-клиента. В ответе сервера может быть указан тип содержимого (изображение, json, html и т.д.) и кодировка. Заголовков много, и прикладной программист может сам создавать новые заголовки. В нашу задачу не входит подробное рассмотрение заголовков, вы можете ознакомиться с ними в документации к протоколу [!!!]. Заголовки отделяются от тела запроса пустой строкой.

На листинге 1.2 и 1.3 приведены примеры клиентского и серверного пакета http. Как видите, это просто специально отформатированный текст. Клиентский пакет называется запросом (request), серверный - ответом (response). Эти слова помогут вам ориентироваться в документации практически любого веб-фреймворка, где Request и Response обычно реализованы в виде отдельных классов.

Листинг 1.2. Пример HTTP request

```
GET /wiki/страница HTTP/1.1
Host: ru.wikipedia.org
User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5
Accept: text/html
Connection: close
```

(пустая строка)

Листинг 1.3. Пример HTTP response

```
HTTP/1.1 200 OK
Date: Wed, 11 Feb 2009 11:20:59 GMT
Server: Apache
X-Powered-By: PHP/5.2.4-2ubuntu5wm1
Last-Modified: Wed, 11 Feb 2009 11:20:59 GMT
Content-Language: ru
Content-Type: text/html; charset=utf-8
Content-Length: 1234
Connection: close
```

(пустая строка)

(запрошенная страница в HTML)

Далее мы рассмотрим необходимые программисту элементы HTTP подробнее.

1.2 HTTP-методы

HTTP-метод – это дополнение к URI в запросе пользователя, которое позволяет понять, запрашивает пользователь данные или отправляет. Есть два основных метода HTTP, которые поддерживают все браузеры, это GET и POST.

GET – самый распространенный метод, с помощью него клиенты запрашивают у сервера ресурсы на чтение. Этот запрос может содержать параметры, но все они указываются в URI. Указать параметры в теле GET-запроса нельзя, поэтому размер данных, которые клиент может передать серверу через этот тип запроса, ограничен. Предполагается, что запрос GET служит для получения ресурсов, и не вызывает изменений (например, создания и удаления страниц) на сервере.

POST – это основной метод, предназначенный для отправки данных на сервер. В этом случае параметры отправляются в теле запроса, что позволяет отправить большое количество данных, в том числе файлы. В простейшем случае этот запрос служит для отправки форм. Также POST принято использовать для запросов, меняющих состояние сервера (добавления-удаления ресурсов, изменения настроек и т.д.) и запуска удаленных процедур.

Строго говоря, никто не запрещает серверу обрабатывать GET-запрос как запрос на изменение (например, можно отправлять форму на создание запроса в разделе FAQ сайта или запустить операцию построения поисковых индексов через GET), но это считается плохим тоном.

В таблице 1.1 приведены примеры распространенных HTTP-методов. Обратите внимание, что не все браузеры поддерживают всё разнообразие методов.

Для одного и того же URI может быть доступно несколько HTTP-методов, это позволяет сделать API веб-сайта более лаконичным. Например, для страницы товара `http://www.mysupershop.ru/product/123` покупателю может быть доступен только GET-запрос, а администратору сайта дополнительно PUT и DELETE.

Мы рассмотрим подробнее, какой метод в каком случае выбирать, в разделе 1.5.

Таблица 1.1 Некоторые HTTP-методы

Метод	Пояснение
GET	запрос содержимого указанного ресурса
POST	передача пользовательских данных заданному ресурсу (через формы, в том числе отправка файлов)
PUT	загрузка содержимого запроса на указанный в запросе URI
DELETE	удаление указанного ресурса
PATCH	это PUT, но применяется только к фрагменту ресурса
TRACE	возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавляют или изменяют в запросе
OPTIONS	определение возможностей веб-сервера или параметров соединения для конкретного ресурса, например, список доступных для данного URI HTTP-методов
CONNECT	преобразует соединение запроса в прозрачный TCP/IP туннель, для защищенного SSL соединения через нешифрованный прокси
HEAD	это GET без тела в ответе сервера, для извлечения метаданных, проверки наличия ресурса, например, чтобы узнать, не изменился ли он с момента последнего обращения

1.3 Коды состояния HTTP

Код состояния HTTP – это обязательная часть ответа сервера. Код состояния представляет собой трехзначное число, которое кратко описывает результат обработки запроса. Можно выделить пять типов кодов состояния, которые описаны в таблице 1.2. Обратите внимание, что тут приведены не все возможные коды состояния, а только примеры для каждой группы. С полным списком вы можете ознакомиться в документации протокола.

Таблица 1.2 Коды состояния

Первая цифра кода	Назначение	Пример
1	Информационные	102 Processing
2	Успех	200 OK – признак успешного получения страницы 201 Created 204 No Content

3	Перенаправление	301 Moved Permanently 302 Moved Temporarily
4	Ошибка на клиенте	400 Bad Request 401 Unauthorized 402 Payment Required 403 Forbidden 404 Not Found 405 Method Not Allowed 418 I'm a teapot
5	Ошибка на сервере	500 Internal Server Error

Коды, которые начинаются с цифры 1 - это различные технические информационные сообщения, на которых мы не будем останавливаться. Подробнее о них вы можете прочитать в документации к протоколу.

Коды, которые начинаются с 2 означают успешную обработку запроса и успешную передачу данных. Наиболее распространенный ответ из этой группы – код 200, означающий штатный успешный ответ на запрос. Вы можете уточнить для клиента результат операции, например, еще в коде состояния указать, что в ответе отсутствует контент (204) или при обработке запроса на создание какого-либо объекта, что он успешно создан (201).

Коды состояния, начинающиеся с 3 – это различные перенаправления. Если вы переместили какую-то страницу на вашем сайте (например, вы разделили категорию товаров «Обувь» на детскую, женскую и мужскую, и у товаров в этой категории поменялись URI), то на старый URI такой страницы рекомендуется создавать перенаправление с кодом 301, которое указывает клиенту, что ресурс навсегда перемещен на другой URI. Это помогает не потерять тех пользователей, которые создали закладку на перемещенную страницу, и улучшает поисковую индексацию после таких изменений.

Коды состояния, начинающиеся с 4, сообщают клиенту о том, что на стороне клиента обнаружена ошибка. Самый известный из кодов этой группы – 404, говорящий клиенту, что он пытается обратиться к несуществующей странице. Хорошим тоном считается оформить страницу для этого кода состояния в дизайне сайта и разместить на ней элементы, которые помогут пользователю сориентироваться, например, главное меню. Коды 401 и 403 означают, что у клиента нет прав доступа к запрашиваемому ресурсу или операции. Код 401 более точно характеризует проблему, сообщая о том, что клиент не авторизован. Код 403 также может быть получен для неавторизованного клиента, но этот код может использоваться и в том случае, когда клиент авторизован, но прав на операцию (например, чтение страницы в административной части сайта или изменение настроек) у него нет. В лабораторных вы можете столкнуться с ошибкой 405, которая означает, что у вас в запросе указан недопустимый для данного URI метод HTTP.

Коды состояния, начинающиеся с 5, означают проблему на сервере. В лабораторных работах это означает, что скорее всего в коде сервера есть ошибка, начиная от неправильных настроек базы и заканчивая ошибкой компиляции. Если вы видите ошибку 500, то либо в среде разработке, либо в логах сервера, либо прямо в ответе сервера в браузере (если ваш сервер запущен в режиме отладки) вы можете найти сообщение о том, что конкретно произошло. В реальных серверах, запущенных в эксплуатацию, необходимо позаботиться о том, чтобы подробности ошибки 500 не были доступны обычным пользователям, так как некоторые серверы по умолчанию выводят на такие страницы свою версию, версию операционной системы и другие критичные данные.

1.4 Сессии и Cookies

Ещё одной важной технологией, о которой должен знать каждый программист, столкнувшийся с web-разработкой, это cookies. Хотя TCP протокол поддерживает возможность создания постоянного канала между клиентом и сервером, HTTP-протокол устанавливает отдельную TCP-сессию на каждый запрос пользователя. Это значит, что сервер, когда он получает запрос от клиента, строго говоря, не знает появлялся ли этот клиент ранее или он пришел впервые. То есть, сервер не знает, какие действия выполнял клиент ранее. Это не очень удобно для приложений, поэтому большинство браузеров поддерживает cookies.

Cookies - это небольшие фрагменты данных, которые отправляются с веб-сервера клиенту и хранятся на компьютере клиента. Веб-клиент всякий раз при попытке открыть страницу или какой-то другой ресурс соответствующего сайта пересылает этот фрагмент данных на веб-сервер в заголовках.

Какие именно cookies надо отправить данному серверу, определяется по доменному имени ресурса. Это значит, в частности, что cookies распространяются на все вложенные домены.

Рассмотрим пример. Домен электронной библиотеки АлтГТУ elib.altstu.ru будет получать не только свои cookies, но и cookies для основного сайта АлтГТУ altstu.ru. При этом запрос к домену elib.altstu.ru не будет содержать cookies для домена yandex.ru, а запрос к домену altstu.ru не получит cookies для elib.altstu.ru.

Приведем основные рекомендации по использованию cookies для программистов.

1. Не делайте cookies многочисленными и не складывайте в них большие объемы информации. Помните, что вся эта информация передается по сети каждый раз, когда клиент запрашивает что-то с вашего сервера, то есть, может существенно замедлить работу системы.
2. Не складывайте в cookies критичную для безопасности информацию, вроде логина или пароля. Cookies можно установить и изменить на клиенте без участия сервера. Кроме того, они могут быть перехвачены, и использованы для атаки на сервер.

3. Вы можете хранить информацию о клиенте, используя идентификаторы сессий. **Идентификатор сессии** – это некоторый уникальный идентификатор, который выдается пользователю при первом обращении на сервер, и затем записывается в cookies. После этого на сервере можно хранить текущее состояние клиента (например, под каким логином он авторизован, какие страницы посещал, какие товары добавил в корзину). При этом информация хранится только на сервере, а на клиенте остается только идентификатор сессии. Сессии могут быть созданы и для неавторизованных пользователей.
4. Cookies и сессии могут иметь время жизни, после которого становятся недействительными.

На рисунке 1.2 приведен один из классических сценариев использования сессий и cookies – авторизация на сайте. Браузер пересылает серверу введенный пользователем логин и пароль. Сервер, если все в порядке, подтверждает, что все хорошо и передает в cookies браузеру идентификатор сессии. Когда браузер осуществляет все остальные запросы, например, запрашивает страницу профиля или пытается загрузить новую аватарку, то он отправляет идентификатор сессии, и по нему уже сервер определяет, что это тот самый пользователь, который заходил ранее.

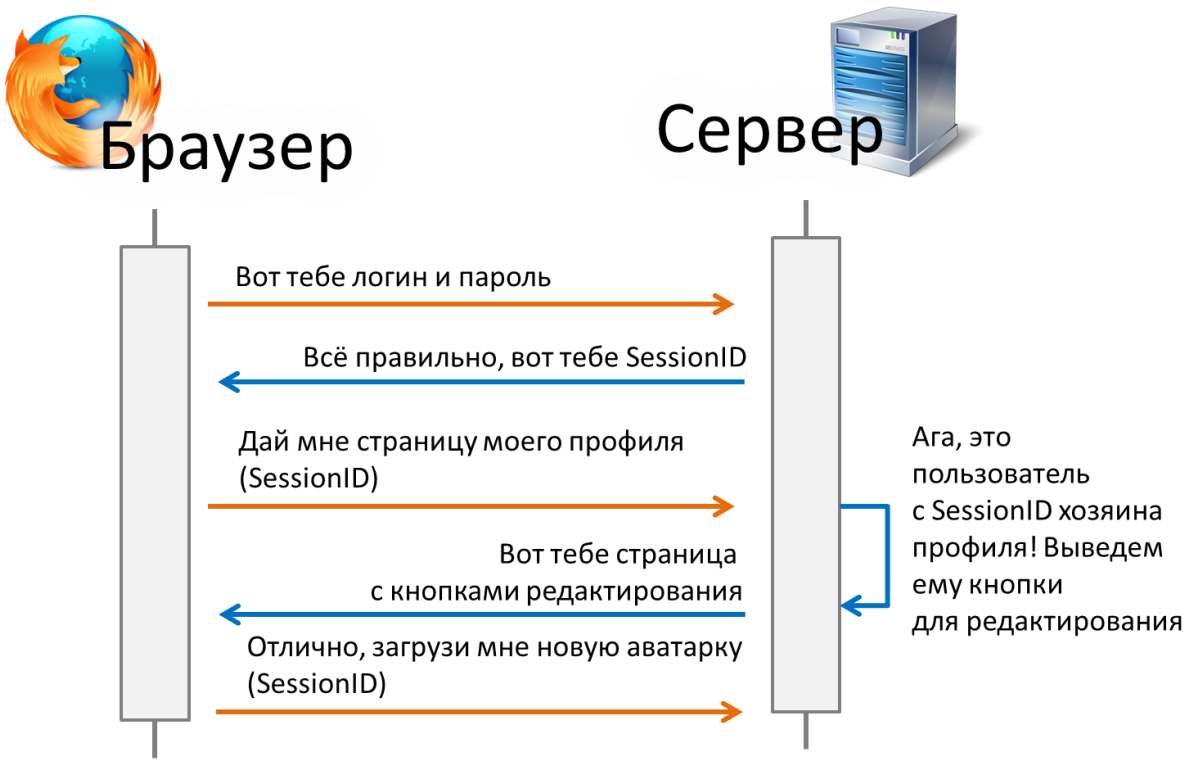


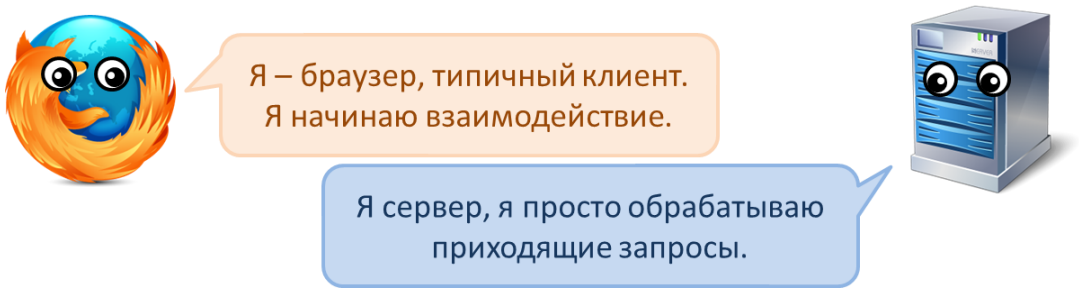
Рисунок 1.2 Пример сценария использования cookies и сессий

1.5 Стандарт REST

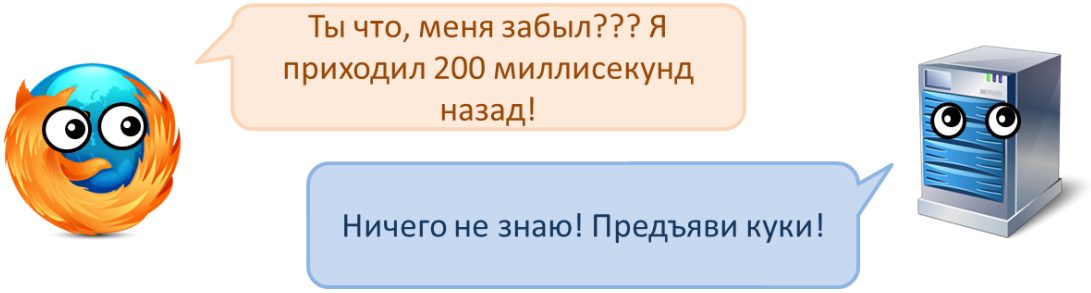
В этом разделе мы рассмотрим одну из базовых концепций функционирования интернета - REST. Хотя данная концепция лежит в самой основе всемирной паутины, это название было предложено совсем недавно, в 2000 году Роем Филдингом, одним из разработчиков протокола HTTP. REST расшифровываться как Representational State Transfer, то есть передача репрезентативного состояния. Это метод взаимодействия компонентов распределенного приложения в интернете при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос. Необходимые данные передаются в качестве параметров этого запроса. Таким образом REST подводит теоретическую базу под лучшие практики разработки интернет-приложений, выработанные за многие годы функционирования всемирной паутины.

REST предъявляет к приложению 5 основных требований.

1. Приложение должно быть реализовано **на основе клиент-серверной архитектуры**.



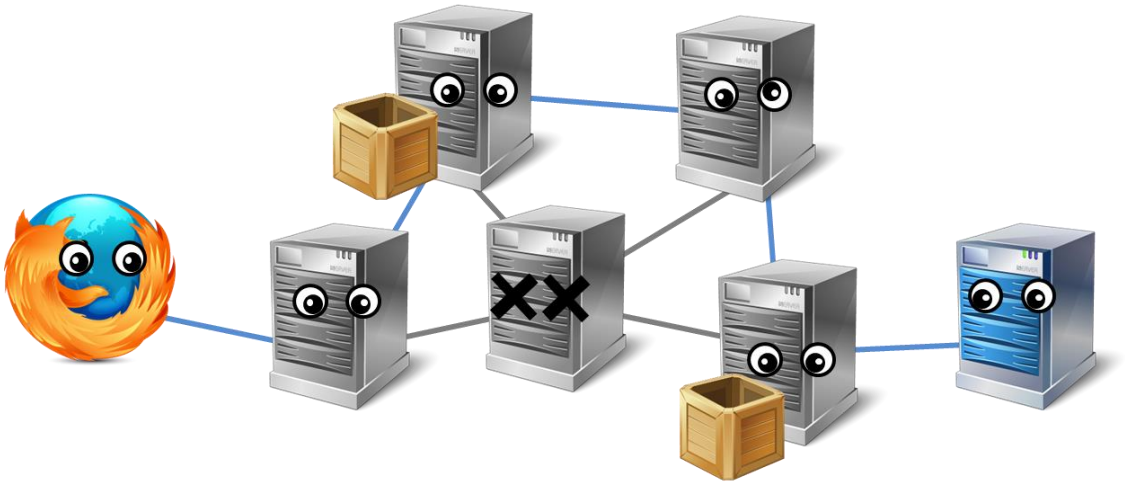
2. **Сервер не обязан хранить информацию о состоянии клиента**. Сохранение состояния сессий клиента полностью отдается на откуп самому клиенту. Во-первых, это повышает надежность системы в случае частичных сбоев, когда на сервере может теряться состояние клиента. Во-вторых, такой подход повышает масштабируемость системы так как не нужно выделять дополнительное место на сервере для хранения этих самых состояний.



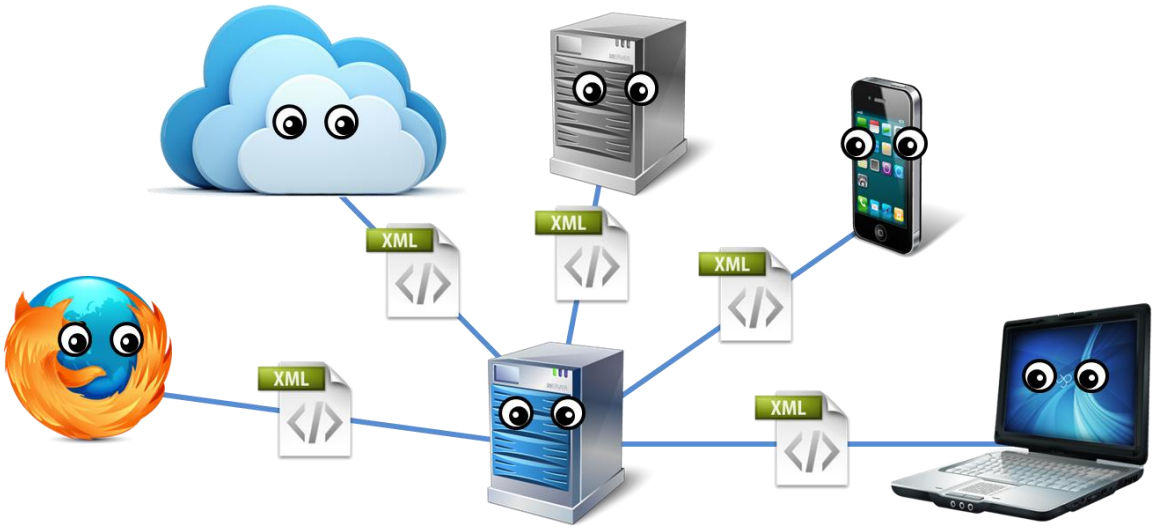
3. **Данные могут кэшироваться**. Кэш - это специальное промежуточное хранилище, которое используется для ускорения доступа к данным. Кэш может находиться как на клиенте, так и на сервере, и на промежуточных узлах.

Поэтому в каждом запросе клиента и в каждом ответе сервера должно быть явно указано можно ли кэшировать данный ответ, и можно ли получать результаты данного запроса из кэша. Самым простым классическим примером такого кэширования является кэширование статических ресурсов, то есть таблиц css, JavaScript-файлов, шрифтов и некоторых картинок. Если в процессе выполнения лабораторной работы вы внесли изменения в статические файлы, например, в css, вы можете не сразу увидеть изменения из-за кэша в браузере. Чтобы получить свежую страницу, нажмите ctrl+F5.

4. Клиент может взаимодействовать не напрямую с сервером, а с произвольным количеством промежуточных узлов. Клиент может не знать о существовании промежуточных узлов, кроме случаев передачи конфиденциальной информации. Это позволяет масштабировать приложения на архитектуру интернета, то есть клиент и сервер могут обмениваться пакетами не непосредственно, а через набор промежуточных узлов. Каждый из этих узлов может обладать своим собственным кэшем, но для клиента и сервера это будет прозрачно, то есть они могут полагать, что взаимодействуют непосредственно друг с другом.



5. Сервис должен обладать универсальным интерфейсом. Это комплексное требование, которое позволяет каждому сервису развиваться независимо. В частности, это означает, что сервис использует универсальные форматы представления информации, такие как HTML, XML или JSON, которые не зависят от способа хранения информации внутри сервера, то есть представление информации отделено от способа его хранения. Мы не будем здесь подробно рассматривать остальные аспекты этого принципа, но, если вы собираетесь заниматься разработкой REST-сервисов, рекомендуем вам ознакомиться с этим принципом самостоятельно.



Итак, подведем итог. Необходимыми условиями REST являются:

- Клиент-серверная архитектура
- Отсутствие хранимого на сервере состояния клиента
- Кэширование
- Возможность наличия промежуточных узлов
- Единообразие интерфейса

Перечислим преимущества данного подхода.

- надёжность (за счет отсутствия необходимости сохранять информацию о состоянии клиента);
- производительность (за счет использования кэша);
- масштабируемость;
- прозрачность системы взаимодействия;
- простота интерфейсов;
- портативность компонентов;
- легкость внесения изменений;
- способность эволюционировать.

Рассмотрим пример разработки REST-интерфейса с использованием HTTP-глаголов: спроектируем REST-интерфейс для просмотра и редактирования списка авторов в электронной библиотеке. Стандартный набор операций для этой задачи называется CRUD (create, read, update, delete). В соответствии с устройством протокола HTTP и принципами REST, URI обозначают ресурсы. В нашем случае, это список авторов и отдельный автор. Пусть за список у нас отвечает URI `/eum/author`, а за отдельных авторов – набор URI `/eum/author/{id}`. Здесь `{id}` – идентификатор автора. Такие шаблонизированные URI вместе с одним из HTTP-методов называются маршрутами.

Методы HTTP в терминологии REST иногда называются REST-глаголами. Философия REST предполагает, что в ваших URI не содержится глагола (например, `POST /eum/author/add` - неудачный маршрут). В таблице 1.3 приведен пример правильного использования HTTP-методов для создания маршрутов, а также соответствие методов операциям CRUD и SQL.

Таблица 1.3 Соответствия между операциями CRUD, REST-глаголами и SQL-операциями

REST	CRUD	SQL	Пример
POST PUT	CREATE	INSERT	`/eum/author` – создать нового автора (POST) `/eum/author/701` – создать автора с номером 701 (PUT)
GET	READ	SELECT	`/eum/author` – получить список авторов `/eum/author/701` – получить материалы автора 701
PUT	UPDATE	UPDATE	`/eum/author/701` – изменить информацию об авторе 701
DELETE	DELETE	DELETE	`/eum/author` – удалить всех авторов `/eum/author/701` – удалить автора с номером 701

1.6 HTTP-сервера

Способ, которым надо обработать запрос, определяется сервером на основе стартовой строки и заголовков. Как уже говорилось, на одной машине и на одном порту может находиться несколько прикладных приложений для http. Кроме того, аккуратное формирование структуры http-пакета и передача по сети - не самая простая задача. Часто ее реализация перекладывается на специальный модуль обработки http-запросов (например, веб-сервер Apache для php или контейнер сервлетов Tomcat для java). Основная задача таких модулей - это получить запрос от клиента и передать его на обработку в определенный скрипт или программу, а потом - получить ответ от этой программы и отправить назад в браузер или в другой клиент. В этом случае часть работы по определению, какой код должен обрабатывать тот или иной запрос, берет на себя этот модуль. Это делается на основе внутренней системы маршрутизации. Обычно это таблица соответствия между доменными именами и программами на сервере. Также обработчики http-запросов как правило берут на себя задачу отправки клиенту некоторых редко изменяемых файлов (картинок, стилей css и скриптов javascript). Такие файлы называются статическими ресурсами.

На рисунке 1.3 приведена статистика популярности самых распространенных веб-серверов.

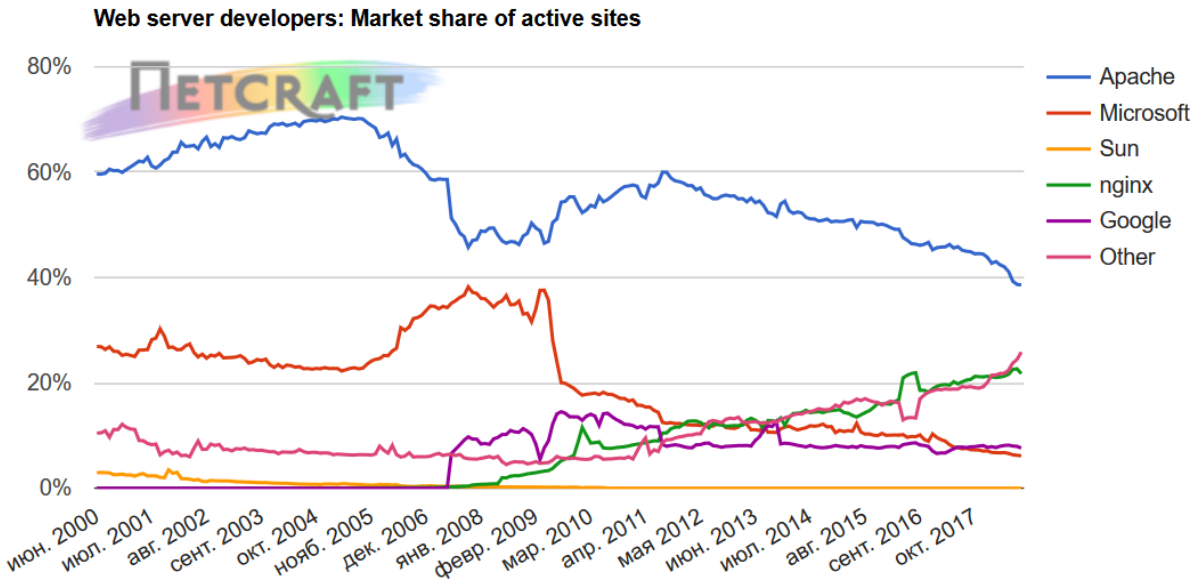


Рисунок 1.3 Популярность веб-серверов[1]

В большинстве веб-фреймворков, которые мы будем рассматривать в этом курсе, есть встроенные веб-сервера, предназначенные для разработки. Использовать их для эксплуатации реальных приложений не стоит, так как они значительно медленнее и потенциально менее безопасные, чем специально разработанные сервера, типа ngnix и apache.

Итак, **HTTP-сервер** - это программа, которая обрабатывает GET и POST запросы по протоколу HTTP. Два наиболее распространенных HTTP-сервера - это Apache и Nginx. Оба сервера могут обрабатывать запросы к статическим ресурсам (страницам в формате html, файлам css, js и картинкам), а также имеют подключаемые модули, которые позволяют использовать скриптовые языки (такие как PHP, Python, Ruby и т.д.) и генерировать динамические html-страницы "на лету" в зависимости от запроса пользователя.

В большинстве случаев современные web-сервера - это не отдельные компьютеры, а виртуальные машины, предоставляемые на условия IaaS или PaaS. Зачастую настройки на реальном сервере существенно отличаются от настроек на локальной машине. Поэтому для разработки может быть использована специальная виртуальная машина с настройками реального сервера. Это позволяет унифицировать среду для всех разработчиков в пределах проекта и избежать проблем с совместимостью. Однако, вы можете избрать другой путь, и устанавливать на своей машине сервера по мере необходимости.

[1] https://news.netcraft.com/archives/2018/09/24/september-2018-web-server-survey.html

Для доступа к веб-серверу используются, как правило, протоколы ftp (file transport protocol) и ssh. SSH - сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов). Чтобы использовать ssh, необходимо установить специальную программу - ssh-клиент.

На практике http-клиентом может выступать не только браузер, но и любое другое приложение, например, бот поисковой системы или мобильное приложение. Точно так же http-сервером может являться любая программа, умеющая обслуживать соответствующие соединения. Нет никаких специальных ограничений на средства разработки таких программ, они могут быть написаны на любом языке программирования общего назначения. Некоторые из языков (например, PHP, JavaScript, Python, Java) используются чаще. Это связано стем, что эти языки имеют богатые библиотеки и фреймворки, а некоторые изначально специально разрабатывались для того, чтобы сделать написание приложений для интернета более быстрым и удобным.

1.7 Инструменты веб-разработки

Использование клиент-серверной архитектуры при обработке http-запросов дает возможность разделить логику работы между сервером и браузером. Поэтому все инструменты веб-разработки делятся на две большие группы. Это frontend-инструменты которые функционируют внутри браузера, и backend-инструменты, которые функционируют и на стороне сервера.

Frontend-инструменты - это в первую очередь html css и javascript. HTML - это язык разметки гипертекста, основной способ представления информации в Web. CSS - язык описания стилей для html-страниц, разработан как надстройка над HTML, позволяющая эффективно управлять такими параметрами, как шрифт и цвет, а также положение элементов страницы относительно друг друга. Javascript - это стандартный скриптовый язык браузеров.

На backend может быть запущена программа на любом языке программирования, но как правило правило это php, java, Ruby, Python, asp.net. Серверная часть может быть написана на C++ или даже на Ассемблере. Данные по HTTP могут передаваться не только в формате html, для которого изначально разрабатывалась инфраструктура web, но и в других форматах, в первую очередь xml и json.

1.8 Адаптивная верстка

Наиболее распространенным способом взаимодействия с пользователем в web являются html-страницы. Верстка HTML-страницы - это превращение макета, как правило из формата Photoshop, то есть psd, в набор файлов, которые являются

заготовкой пользовательского интерфейса сайта. Результатом верстки является шаблон. На этом этапе еще не реализована вся логика сайта: не загружаются данные из хранилища, не обрабатываются различные сообщения, - но могут работать некоторые интерактивные элементы, например галереи. В html-шаблон входят html-файлы для различных видов страниц (для одного типа страниц, например, карточки товара, делается одна html-страница, а для списка товаров нужна другая страница), javascript-файлы, css-стили и набор необходимых картинок.

Верстка сложных шаблонов - довольно трудоемкий процесс, ведь результат необходимо проверить в нескольких браузерах. Несмотря на то что большинство современных браузеров неплохо поддерживают имеющиеся стандарты, все равно между ними существуют различия. Во-первых, это различный стиль по умолчанию, когда у вас вдруг обнаруживается, что какой-нибудь из браузеров рисует некрасивую рамочку вокруг текущего элемента формы, не подходящую по цвету к общему оформлению сайта. Во-вторых, умение автомасштабировать элементы и изображения, по-разному выравнивать положение текста и создавать внезапные горизонтальные полосы прокрутки на различных разрешениях экрана, в первую очередь на мобильных устройствах. В-третьих до сих пор некоторые браузеры имеют нестандартные возможности и атрибуты, которые могут не поддерживаться в других браузерах. Браузеры по-разному поддерживают события мыши и события касания для мобильных версий.

Кроме того, всегда проблемой верстальщика будет разрастающейся зоопарк устройств. Как правило, поддерживаются размеры экрана от 320 пикселей в ширину и практически до бесконечности. Обычно все устройства делят на четыре группы:

- мобильные телефоны;
- планшеты;
- ноутбуки;
- широкоформатные стационарные мониторы.

На каждом из этих мониторов сайт должен выглядеть как минимум приемлемо. Если сайт на устройстве выглядит коряво, пользователи просто уходят с него, и владелец сайта начинает терять прибыль.

Развитие мобильных устройств привело к тому, что от современных интернет-страниц требуется корректное отображение не только в разных браузерах, но и в очень большом диапазоне расширений экрана. Верстка, удовлетворяющая этому требованию, называется **отзывчивой**. Отзывчивая верстка - это компонент адаптивной, различия между этими понятиями вы можете найти в Википедии (см. список литературы). Такая верстка требует тщательного тестирования и учета множества нюансов. В частности, адаптивная верстка подразумевает подход mobile first. Он означает, что страница верстается изначально под мобильные устройства, и только потом дорабатывается для более крупных экранов.

Для реализации адаптивной верстке в стандарте CSS 3 появились специальные условия применения правил, которые могут задавать размер экрана, для которого правило действует. Пример использования такого условия приведен на листинге 1.4

Листинг 1.4. Пример правила с ограничением на минимальную ширину экрана, для которой оно применяется

```
@media (min-width: 768px) {
  .hereIsMyClass {
    width: 30%;
    float: right;
  }
}
```

Производство сайтов - огромная индустрия, и создавать сложную адаптивную верстку с "нуля" невыгодно. В качестве основы для будущих интернет-страниц используются специальные css-фреймворки. Два наиболее популярных из них – это Twitter Bootstrap и Zurb Foundation. Css-фреймворк состоит из одного css-файла (стили страницы) и одного или нескольких js-файлов (скрипты), и обладает широкими возможностями: качественная адаптивная верстка на основе сетки (попробуйте изменить размер окна браузера на официальном сайте фреймворка), большое количество разнообразных элементов (формы, панели навигации, слайдеры, маркеры, всплывающие окна и многое другое).

Обычно CSS-фреймворк включает в себя следующие компоненты.

- Стили типографики для заголовков, абзацев, списков, таблиц и т.д.
- Стили форм для полей ввода, кнопок, выпадающих списков и проч., в том числе для создания форм в одной строке (например, для компактной формы поиска).
- Сетка – способ управления положением элементов на странице. Некоторые фреймворки могут поддерживать несколько видов сеток. Как правило, сетка позволяет создавать блоки-строки и блоки-колонки. Для колонок может быть задана ширина для различных разрешений экрана. Кроме того, для любого элемента может быть задано, на каких размерах экрана он отображается, а на каких его надо скрыть.
- Компоненты:
    - навигация (панели для верхнего, нижнего и бокового меню);
    - вкладки (tabs), аккордеоны;
    - слайдеры, галереи;

- всплывашки (popup-окна) и сообщения (alerts);
- иконки;
- и т.д.

При создании страницы на основе этого фреймворка создается html-код (широкий спектр примеров находится на сайте фреймворка) и дополнительный файл css, который позволяет настроить верстку под конкретный дизайнерский макет.

СУБД MySQL

MySQL - свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией.

Существует множество инструментов для администрирования MySQL, как платных, так и бесплатных. Наиболее распространенный среди них - phpMyAdmin, бесплатное php-приложение, которое, как правило, установлено на серверах интернет-провайдеров. Аналогом является приложение Adminer.

MySQL поддерживает несколько типов таблиц, что позволяет гибко настраивать базу в зависимости от соотношений операций чтения-записи, необходимого уровня скорости работы и надежности. Рассмотрим два наиболее используемых типа таблиц. Другие типы таблиц вы можете изучить в документации (см. ссылку в списке литературы).

MyISAM -Данный тип таблиц предназначен для быстрого доступа к информации. Исторически - один из ранних типов таблиц в MySQL. Многие недостатки не позволяют использовать этот тип таблиц в больших проектах, но для блога, сайта-визитки и других маленьких и средних проектов, не требующих большого количества операций записи и не хранящих критическую информацию, данный вариант наиболее эффективен.

- транзакций нет
- макс. размер на диске: 256ТБ
- блокировка: вся таблица
- возможность полнотекстового индекса и быстрого поиска
- отсутствует поддержка целостности и внешних ключей
- поддержка репликации
- каждый столбец может иметь свою кодировку
- медленные операции записи (не подходит для высоконагруженных систем)

- невысокая надежность (не используйте для финансовых операций и делайте бэкапы!)

InnoDB - один из наиболее быстрых и надежных современных движков для таблиц (не только в MySQL). Операции чтения работают значительно медленнее, чем в MyISAM, особенно если отсутствуют индексы.

- полная поддержка транзакций (4 уровня изоляции)
- макс. диск: 64ТБ
- блокировка записи (не таблицы), два вида блокировок (SHARED, EXCLUSIVE)
- полнотекстовый индекс: нет
- индексы кластеризуются для «типичных запросов»
- поддержка целостности (внешние ключи)

Контрольные вопросы

1. Что такое клиент-серверная архитектура?
2. Как устроен протокол HTTP?
3. Что такое URI?
4. Как устроен пакет HTTP? Какие два типа пакетов вы знаете?
5. В чем разница между клиентским и серверным пакетом HTTP?
6. Что такое стартовая строка HTTP-пакета?
7. Что такое заголовки HTTP? Какие заголовки вы знаете?
8. Что такое метод HTTP? Какие методы вы знаете? В чем их отличие?
9. Что такое код состояния? Сколько типов кодов состояния вы знаете?
10. Что означают распространенные коды состояния: 200, 404, 500? Какие еще коды состояния вы знаете?
11. На основе чего определяется, какое приложение должно обработать запрос?
12. Что такое сессии в HTTP?
13. Что такое куки и зачем они нужны?
14. Что такое браузер?
15. Что такое веб-сервер?
16. Какие веб-сервера вы знаете?
17. Что такое фронтенд и бэкенд? Приведите примеры инструментов для реализации функционала фронтенда и бэкенда?
18. Что такое REST?
19. Перечислите принципы REST.
20. Что такое кэширование?
21. Что такое адаптивная верстка? Зачем она нужна?

22. Что такое css-фреймворк?

Лабораторная работа 1

Цели работы

Цель лабораторной работы - познакомиться с веб-серверами, создать заготовки HTML-шаблонов и разработать схему БД для остальных лабораторных.

Задание

Лабораторная работа состоит из трех обязательных частей.

- Установка инфраструктуры **обязательно**
- Создание базы MySQL **обязательно**
- Адаптивная верстка Twitter Bootstrap или Zurb Foundation **обязательно**

Сдавать каждую часть можно отдельно. Все эти части являются необходимым допуском к остальным лабораторным работам.

Кратко, что надо сделать:

1. Установить пакет серверов XAMPP[2].
2. Изучить инструменты администрирования для MySQL и создать базу данных для последующих лабораторных работ.
3. Создать адаптивный html-шаблон на основе одного из css-фреймворков для использования в последующих лабораторных работах.

Настраивать вывод в шаблоны информации из базы данных не нужно, этому посвящены следующие лабораторные работы.

Ссылки для скачивания инструментов

- Набор серверов XAMPP https://www.apachefriends.org/ru/index.html
- Twitter Bootstrap http://getbootstrap.com/
- Zurb Foundation https://foundation.zurb.com/

Требования к результатам и отчету

Требования к базе данных

- База развернута на сервере MySQL и студент умеет получать к ней доступ с помощью phpMyAdmin
- В базе имеется 3 и более связанных таблиц
- Таблицы поддерживают русскую кодировку utf8

[2] Вы можете использовать другие пакеты серверов или отдельные веб-сервера. Главное требование – наличие доступа по доменному имени к свёрстанным в пункте 3 шаблонам.

- В таблицах реализованы автоинкрементные первичные ключи
- В таблицах настроены внешние ключи
- Таблицы заполнены данными, которые можно пополнять, изменять и удалять

Требования к HTML-шаблонам

- Имеется 4 шаблона:
    1. Страница для отображения списка основных объектов клиенту сайта
    2. Страница для подробного просмотра одного объекта клиентом сайта
    3. Страница для просмотра списка объектов администратором сайта (с кнопками удаления, добавления и редактирования, может быть объединена с первым шаблоном)
    4. Страница для редактирования объекта
- **Важно!** Обратите внимание, мне не нужна ваша страница с красивым слайдером или еще какая-то такая ерунда, мне нужен список объектов и страницы для просмотра и редактирования объектов. Не тратьте время зря, эти страницы не засчитываются.
- Верстка валидна (не содержит грубых нарушений стандарта HTML5)
- Файлы верстки реализованы в кодировке utf8
- Реализованные шаблоны используют Twitter Bootstrap 4, Foundation 6 или другой фреймворк для адаптивной верстки. *Лайфхак:* как показывает практика, всякие новомодные инструменты "быстрой" визуальной верстки работают не очень хорошо. Так что лучше руками накопипатить из документации фреймворков.
- Верстка адаптивная, и студент может это продемонстрировать с помощью инструментов браузера. *Лайфхак:* все браузеры умеют переходить в режим мобильного устройства, особенно круто это делает Chrome; найдите, как это включается.
- На сайте есть навигационная панель, и она ведет себя хорошо на разных разрешениях экрана
- Сайт обладает понятным и удобным интерфейсом, в соответствии с задачей. Простецкая таблица а-ля phpMyAdmin не пойдет. Куча гигантских картинок, когда на экран при просмотре списка помещается полтора объекта, тоже. *Лайфхак:* во фреймворках есть готовые решения для карточек товаров, и вообще всяких списков, используйте их.
- На сайте реализована кастомизация стилей: изменены шрифты, поля, цвета по сравнению со стандартной схемой Twitter Bootstrap или Foundation, добавлены фоновые изображения.
- **Дополнительно:** использован LESS или SASS
- **Дополнительно:** прикручен слайдер (например, при просмотре одного объекта разные группы его характеристик выводятся в слайдере или во вкладках), галерея или другие интерактивные возможности (которые успешно масштабируются при переходе на различные экраны).

Верстка должна быть размещена на веб-сервере и доступна как сайт, то есть по доменному имени на вашем локальном компьютере.

Обратите внимание! Вы создаете шаблон, заготовку сайта! Загружать туда данные из базы не надо, это просто статические страницы.

Требования к отчету

Отчет должен содержать:

- титульный лист,
- задание (краткое общее задание из этих методических указаний и вариант задания, который выполняет студент),
- исходный код вёрстки,
- скриншоты верстки (для различных экранов),
- структуру базы данных,
- код скрипта генерации базы данных.

Код можно печатать шрифтом 10pt с одинарным интервалом.

Порядок выполнения

1 Порядок выполнения: установка XAMPP[3]

Краткая справка

Полный пакет содержит:

- Web-сервер Apache с поддержкой SSL;
- СУБД MySQL;
- PHP (интерпретатор);
- Perl (интерпретатор);
- FTP-сервер FileZilla (файловый сервер);
- POP3/SMTP сервер (почтовый сервер. может быть заменен на заглушку sendmail);
- TomCat сервер (сервер запуска java-сервлетов);
- утилиту phpMyAdmin (администратор СУБД MySQL на PHP).

При установке вы можете не ставить Perl, FileZilla, TomCat. Они не потребуются для лабораторных.

XAMPP свободно распространяется согласно лицензии GNU General Public License и является бесплатным web-сервером, способным обслуживать динамические страницы.

[3] Следование этой инструкции сэкономит ваше время. Вы можете использовать другие пакеты серверов или отдельные веб-сервера. Главное требование – наличие доступа по доменному имени к свёрстанным в пункте 3 шаблонам.

Ядро Apache включает в себя основные функциональные возможности, такие как обработка конфигурационных файлов, протокол HTTP и система загрузки модулей. Ядро (в отличие от модулей) полностью разрабатывается Apache Software Foundation, без участия сторонних программистов. Теоретически, ядро apache может функционировать в чистом виде, без использования модулей. Однако, функциональность такого решения крайне ограничена.

Система конфигурации Apache основана на текстовых конфигурационных файлах. Имеет три условных уровня конфигурации. - Конфигурация сервера (httpd.conf). - Конфигурация виртуального хоста (httpd.conf с версии 2.2, extra/httpd-vhosts.conf). - Конфигурация уровня директории (.htaccess).

Apache имеет собственный язык конфигурационных файлов, основанный на блоках директив. Практически все параметры ядра могут быть изменены через конфигурационные файлы. Большая часть модулей имеет собственные параметры.

Установка

1. Скачать и распаковать пакет XAMPP. При установке отключайте анивирус.
2. Запустить панель управления xampp-control.exe, стартовать сервер Apache и MySQL. Запускать сервера необходимо с административными правами и в доверенной зоне антивирусного ПО. Сервер Apache занимает порт 80 и 8080, и может вступать в конфликт с другим ПО, например, Skype. В случае возникновения проблем используйте навыки поиска в интернете для решения проблемы перед тем, как обратиться к преподавателю. Если преподаватель за один поисковый запрос может решить вашу проблему, вы будете жестоко затроллены.
3. Попробуйте зайти на сайт localhost. Скорее всего, вы увидите ошибку 403 Access forbitten. Если вы увидели нормальную страницу, на которой, в частности, в правом верхнем углу есть ссылка на phpmyadmin, то пропустите пункт 4.
4. Сконфигурировать Apache (не удивляйтесь, если эти строки уже находятся в нужном состоянии, в зависимости от версии XAMPP там уже могут быть установлены эти настройки). В xampp изначально закрыты права доступа к ресурсам. Зайдите в файл xampp\apache\conf\extra\httpd-xampp.conf и исправьте блок в самом конце файла: Замените Require local на Require all granted .
5. <LocationMatch "^/(?i:(?:xampp|security|licenses|phpmyadmin|webalizer|server-status|server-info))">
    a. Require local
    b. ErrorDocument 403 /error/XAMPP_FORBIDDEN.html.var
6. </LocationMatch>
7. **Только для старых версий, в новой вроде не потребовалось** Сконфигурировать хост-файл для Apache. Зайдите в файл xampp\apache\conf\extra\httpd-vhosts.conf (здесь хранится конфигурация сайтов, о ней будет ниже) и добавьте конфигурацию для инструментов xampp:

```
<VirtualHost *:80>
ServerName localhost
DocumentRoot "/Applications/XAMPP/xamppfiles/htdocs"
<Directory "/Applications/XAMPP/xamppfiles/htdocs">
Options Indexes FollowSymLinks Includes execCGI
AllowOverride All
Allow From All
Order Allow,Deny
</Directory>
</VirtualHost>
```

8. Убедиться, что всё заработало, зайдя в браузере http://localhost/. Зайдите по ссылке phpMyAdmin, чтобы убедиться в работе MySQL.
9. Установить уровень отображения ошибок для PHP. Так же, как и в других языках, PHP-интерпретатор умеет обнаруживать несколько видов ошибок. Часть из них - критические, и при их появлении работа скрипта останавливается, а в браузер возвращается код ошибки (обычно 500). Другие ошибки (например, использование устаревшей функции) не вызывают прекращение работы скрипта. При этом подробности ошибок могут выводиться в браузер, могут писаться в лог-файлы или не выводиться вообще. Рекомендуется настроить уровень отображения ошибок PHP следующим образом. Нужно открыть файл C:/xampp/php/php.ini в любом редакторе и в нем поставить значение: error_reporting = E_ALL (в зависимости от версии XAMPP там уже может быть установлена эта настройка). Подробнее про список ошибок можно узнать в документации. После внесения изменений в файл конфигурации PHP (php.ini) так же нужно перезагружать сервер Apache. В контрольной панели XAMPP напротив "Apache" нажать кнопку "Stop", если сервис уже запущен, дальше после остановки нажать кнопку "Start".

Конфигурирование домена

Далее мы добавим наш сайт, расположенный в каталоге "c:/full/path/to/MyDir/web" (вы должны указать вашу папку на диске). Сайт будет доступен по доменному имени mysite.local . **Важно!** Вы можете выбрать любое доменное имя, которое вам нравится, но не оставляйте имя из примера. Выберите название в соответствии с предметной областью.

1. Создать в любом месте на диске, где вам удобнее, папку для нового проекта (для примера будет использоваться название MyDir). Создать внутри папки MyDir папки web для содержимого сайта и logs для хранения логов.
2. Создать в папке web файл index.htm с текстом Hello, world!`.
3. Перейти в файл xampp/apache/conf/extra/httpd-vhosts.conf, отвечающий за настройку виртуальных хостов. Открыть файл в любом текстовом редакторе и добавить новый хост mysite.local (это доменное имя сайта, как yandex.ru):

```
<VirtualHost *:80>
 ServerAdmin myaddress@mysite.local
 DocumentRoot "c:/full/path/to/MyDir/web"
 ServerName mysite.local
 ServerAlias www.mysite.local
 ErrorLog "c:/full/path/to/MyDir/logs/error.log"
 CustomLog "c:/full/path/to/MyDir/logs/access.log" combined
 <Directory "c:/full/path/to/MyDir/web">
  AllowOverride All
  Order allow,deny
  Allow from all
  Require all granted
 </Directory>
</VirtualHost>
```

4. Перезапустить Apache. Добавить в файл c:/Windows/System32/drivers/etc/hosts строки
   `127.0.0.1 mysite.local`
   `127.0.0.1 www.mysite.local`
5. Убедиться, что сайт mysite.local заработал.

2 Порядок выполнения: адаптивная верстка

Ниже приведен порядок действий для Twitter Bootstrap 4. Вы можете сделать всё то же самое для Zurb Foundation 6.

1. Создайте папку для вашего проекта и настройте на него виртуальный хост, если вы не сделали этого при установке XAMPP.
2. Создайте в папку файл index.html. Это будет главный файл вашего шаблона, тут надо разместить макет со списком объектов по заданию.
3. Создайте все остальные файлы шаблонов, необходимые по заданию. Назовите их, как вы считаете нужным.
4. Зайдите на сайт http://getbootstrap.com/. Здесь есть вся необходимая документация с кучей примеров. Да, всё на английском. Пожалуйста, не ищите на русском, вы можете наткнуться на старые версии документации.
5. Зайдите на страницу для скачивания и скачайте CSS-фреймворк (Twitter Bootstrap 4 илиZurb Foundation 6 на выбор). Распакуйте архив в ваш каталог для шаблонов.
6. Если в архиве отсутствует основной шаблон, зайдите на страницу с основным шаблоном на основном сайте фреймворка. Скопируйте шаблон в свой файл index.html. Изучите получившийся код. Сохраните файл. **Помните, что все файлы должны быть в utf-8**.

7. Убедитесь, что все ссылки в тегах ведут по правильным путям. Обратите внимание, что jQuery часто подключается с внешнего ресурса. Если вы не сможете демонстрировать ваш сайт на компьютере с включенным интернетом, скачайте jQuery и измените ссылки. Не пишите абсолютные ссылки в вашей файловой системе. Не пишите и относительные ссылки. Правильно начинать ссылку на веб-ресурс с корня сайта, например, так: `/css/style.css`. То есть, ссылка начинается с символа «/».
8. Изучите примеры работы с сеткой в вашем фреймворке, посмотрите способы создания колонок и как сделать ширину колонок зависимой от размера экрана. Изучите многочисленные примеры для различных стандартных объектов: картинок, таблиц и прочего. **Обратите внимание, для всего есть примеры кода.**
9. Изучите интерактивные возможности фреймворка: создание слайдеров, навигационных панелей, вкладок и прочего.
10. Создайте сетку для вашего главного шаблона, добавьте панель навигации и все необходимое. Проверяйте, как выглядит ваша страница в браузере, пока она не начнет вам нравится. Смело добавляйте необходимые изображения.
11. Создайте свой дополнительный файл стилей, где вы сможете изменять цвета, поля и другие параметры, как вам нравится, а также добавлять любые собственные стили.
12. Покопайтесь в бесплатных шрифтах от Google, найдите тот, который будет вас радовать весь семестр. Подключение шрифтов от Google требует связи с интернетом при демонстрации лабораторной.
13. Используйте инструменты браузера, чтобы проверить, как ваш сайт выглядит на мобильных устройствах.
14. Не забудьте, что надо сделать все макеты по заданию.
15. Добавьте разумные тексты на страницы, чтобы они смотрелись, как настоящие.
16. Порадуйтесь хорошо сделанной работе.

Вы можете сделать все то же самое, но с другим фреймворком, например, w3.css.

3 Порядок выполнения: создание базы MySQL

1. Запустите сервер apache и mysql в панели XAMPP.
2. Зайдите на localhost через браузер.
3. Найдите ссылку на phpMyAdmin.
4. Создайте базу. Не забудьте, что кодировка должна быть `utf8_general_ci`.
5. Создайте необходимые таблицы. Во всех должны быть автоинкрементные первичные ключи.
6. Настройте внешние ключи.
7. Добавьте несколько записей, убедитесь, что все работает.

8. Сделайте экспорт скрипта, для отчета и бэкапа.
9. Порадуйтесь хорошо сделанной работе.

Порядок защиты лабораторной работы

В процессе защиты работы вы должны предоставить:

- Установленный веб-сервер в составе XAMPP с созданным на нем виртуальным хостом.
- Размещенные на хосте 3 или 4 html-файла с версткой по заданию (на сервере также должны быть размещены все необходимые js-скрипты, стили, изображения).
- Размещенную на сервере базу данных в соответствии с заданием. В базе данных должна работать ссылочная целостность и русскоязычная кодировка.

Вам также необходимо предоставить отчет в электронном виде. **Сохраняйте отчеты до конца семестра, архив со всеми отчетами надо будет предоставить на экзамен.**

Будьте готовы, что в процессе защиты вам могут быть заданы вопросы по html-коду, по адаптивной верстке, по MySQL и т.д.

Дополнительная литература и документация

1. Документация по nginx http://nginx.org/ru/
2. Документация по СУБД MySQL https://www.mysql.com/
3. Дополнительная литература. Адаптивный веб-дизайн https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0%D0%BF%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%B2%D0%B5%D0%B1-%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD
4. Дополнительная литература: Статистика популярности веб-серверов https://news.netcraft.com/archives/2016/12/21/december-2016-web-server-survey.html
5. Дополнительная литература: классификация css-фреймворков https://habrahabr.ru/company/mr_gefest/blog/273553/
6. Дополнительная литература: Краткая справка по REST https://ru.wikipedia.org/wiki/REST
7. Дополнительная литература: Подробное сравнение веб-серверов на Википедии https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BE%D0%B2
8. Дополнительная литература: список css-фреймворков с характеристиками на Хабре https://habrahabr.ru/post/156747/