ЛЕКЦИЯ
Web. Asp .Net

Лектор Крамар Ю.М.

Содержание

1. Основы Web
2. Протоколы TCP/IP и HTTP

ОСНОВНЫЕ ПОНЯТИЯ WEB

Основные определения Web

- **Web,** World Wide Web, Всемирная паутина — глобальное информационное пространство, основанное на физической инфраструктуре Интернета и протоколе передачи данных HTTP.
- **Internet,** Интернет, «Инет», «Всемирная сеть», «Глобальная сеть» — всемирная система добровольно объединённых компьютерных сетей, построенная на использовании протокола IP и маршрутизации пакетов данных. Интернет образует всемирную (единую) информационную среду — способ организации оцифрованной информации. Интернет служит физической основой для Всемирной паутины.
- **Web-технологии** — комплекс технических, коммуникационных, программных методов решения задач организации совместной деятельности пользователей с применением сети Интернет.
- **Контент** — информация, размещенная на веб-страницах.
- **URI страницы** – уникальный адрес страницы в сети Интернет.

URI страницы

URI страницы – уникальный адрес страницы в сети Интернет.

Пример: http://www.yandex.ru/all_services.html

URI включает:

1. 1.Метод доступа к ресурсу, т. е. протокол доступа (http).
2. 2.Сетевой адрес ресурса (www.yandex.ru)
3. 3.Полный путь к файлу на сервере (all_services.html).

Кроме термина **URI** существуют еще **URL** и **URN**, и было бы хорошо понимать различия между ними.

URI, URL и URN

1. **URI** — Uniform Resource Indicator (унифицированный идентификатор ресурса)
2. **URL** — Uniform Resource Locator (унифицированный определитель местонахождения ресурса)
3. **URN** — Uniform Resource Name (унифицированное имя ресурса)

Соблюдаются следующие зависимости между URI, URL и URN:

URI = URL или URI = URN или URI = URL + URN

Например:
URI = http://handynotes.ru/2009/09/uri-url-urn.html
URL = http://handynotes.ru
URN = /2009/09/uri-url-urn.html

Web-технологии

Web-технологии как концепция работы с информацией, отличаются следующими особенностями:

- техническая основа web-технологий – локальные и глобальные сети, часто Интернет;
- применение особого типа тонких клиентов: web-браузеров;
- первоначально преимущественно текстовая и статично-графическая подача информации, в дальнейшем – расширение медиаконтентом;
- изменения в информационных источниках мгновенно отражаются в публикациях;
- число потребителей информации практически не ограничено. публикатор сам может задать особые условия на доступ к публикуемой информации;
- в публикациях могут содержаться ссылки на другие публикации без ограничения на местоположение и источники материалов;
- активная работа поисковых машин;
- доставка и тиражирование контента практически бесплатны.

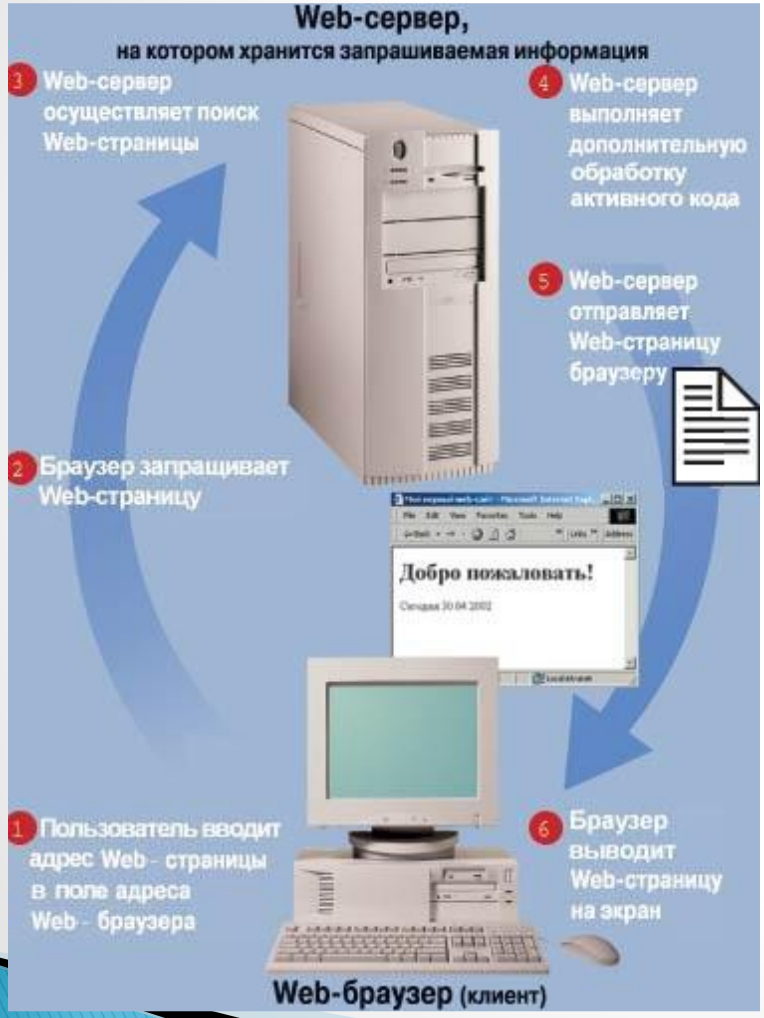
Сайт, страница, сервис, портал

- **Web-страница** (гипертекстовый документ) — документ, описанный на языке HTML. Основное отличие от текстовых документов состоит в том, что он может включать ссылки на другие аналогичные документы. Страницей называют то, что показывает браузер при вводе адреса страницы или при переходе по ссылке.
- **Сайт** (веб-сайт) – совокупность страниц, созданных с применением программного обеспечения, образующая единое целое в техническом, информационном и навигационном аспектах. Чаще всего все страницы сайта имеют общее доменное имя.
- **Сервис** (веб-сервис) – специализированный сайт для решения необходимых посетителям достаточно узких задач. Пример сервиса – gmail.com или www.google.com.

Сайт, страница, сервис, портал

- **Web-сервер** (часто http-сервер) — специальное ПО, установленное на общедоступном компьютере. Web-сервер обеспечивает одновременную обработку запросов от множества клиентов по протоколу HTTP.
- **Портал** – 1. Крупный тематический сайт, активно развиваемый и посещаемый, с четко определенной и растущей аудиторией и постепенно пополняемый сервисами. Примеры: free-lance.ru, www.RussianRealty.ru. 2. Сайт без выраженной тематической направленности или с широкой тематикой (например, географической), имеющий в своем составе множество сервисов, служб и направлений взаимодействия. Примеры: yandex.ru, vk.ru.

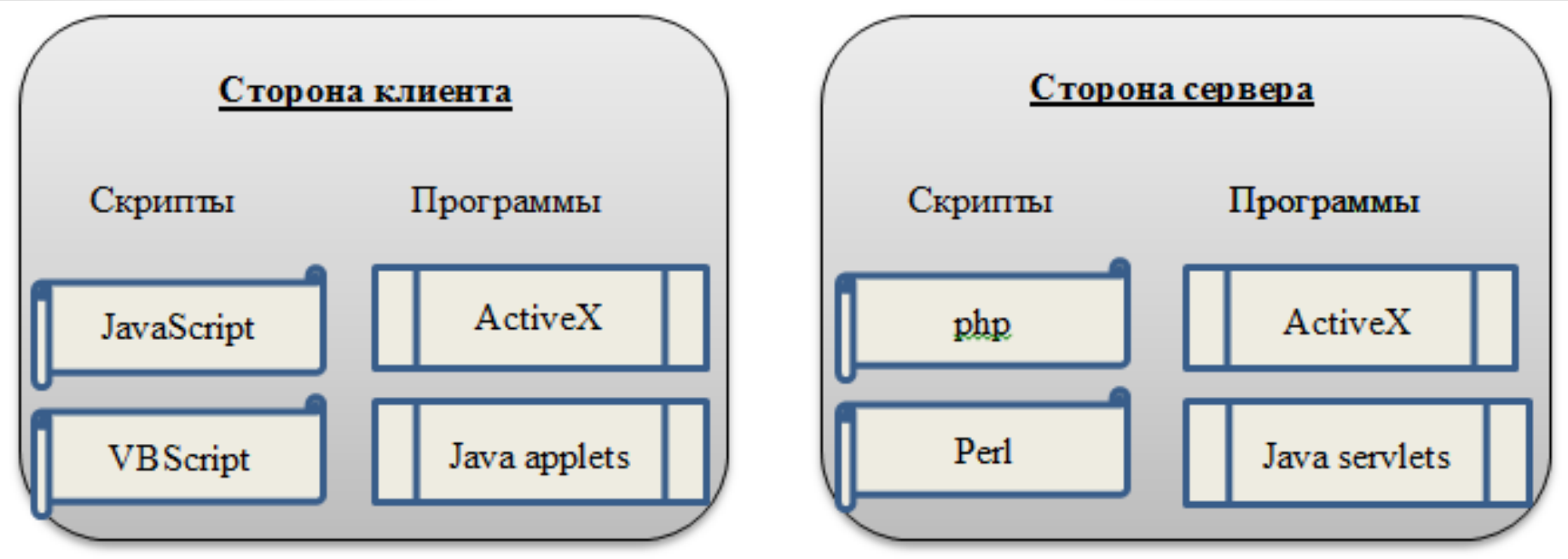
Веб-сервер и браузер, их взаимодействие



Все сайты открываются браузерами как html-документы. Браузер и веб-сервер взаимодействуют по технологии клиент-сервер.

После ввода адреса в строку адреса браузер формирует запрос к серверу. Сервер формирует страницу и передает ее браузеру. Браузер выводит страницу пользователю, который своими действиями формирует новый запрос.

Средства создания динамических страниц



Программное обеспечение для Web-взаимодействия

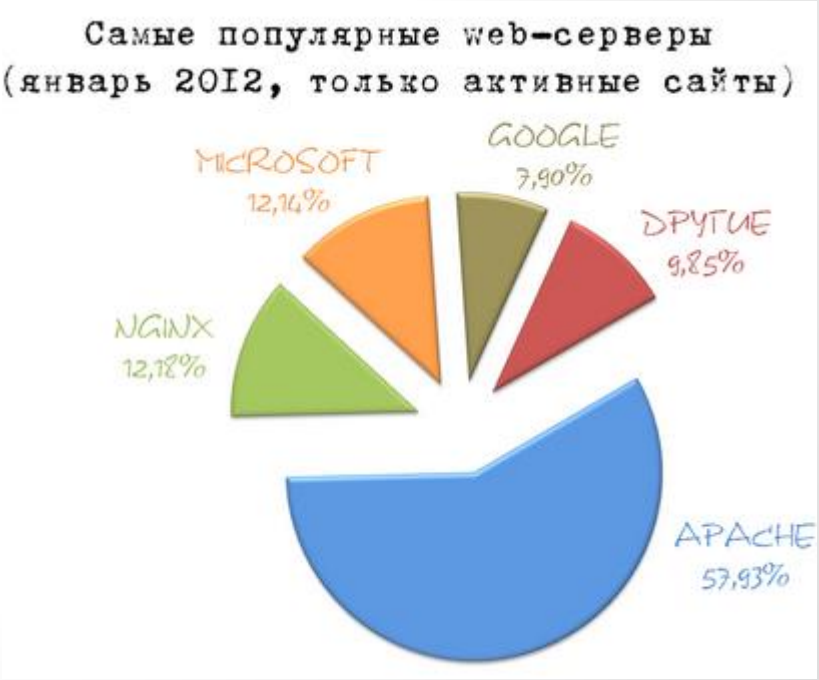
- **Серверы** как основные поставщики услуг хранения и обработки информации (обработка запросов).
- **Клиенты** как конечные потребители услуг сервера (отправка запроса).
- **Прокси** для выполнения транспортных служб.

Для отличия конечных серверов от прокси в официальной документации используется термин «исходный сервер» (*origin server*). Один и тот же программный продукт может одновременно выполнять функции клиента, сервера или посредника в зависимости от поставленных задач.

В спецификациях протокола HTTP подробно описывается поведение для каждой из этих ролей.

Наиболее часто используемое программное обеспечение для Web-взаимодействия

- **Серверы** – Apache, Internet Information Services (IIS), nginx, google и др.
- **Клиенты** – Internet Explorer, Opera, Mozilla Firefox, Netscape Navigator и др.
- **Прокси** – Squid, UserGate, Multiproxy, Naviscope.



ПРОТОКОЛЫ TCP/IP И HTTP

Назначение протокола TCP/IP

TCP (*transmission control protocol* — «протокол управления передачей») — один из основных протоколов передачи данных в сети интернет, предназначенный для управления передачей данных. Сети и подсети, в которых совместно используются протоколы TCP и IP, называются сетями TCP/IP.

TCP работает на 4 (транспортном) уровне между двумя конечными системами, например, браузером и веб-сервером, осуществляя надежную передачу потока байтов от одной программы на некотором компьютере к другой программе на другом компьютере (например, программы для электронной почты, для обмена файлами). TCP контролирует длину сообщения, скорость обмена сообщениями, сетевой трафик.

IP (*Internet Protocol*) – маршрутизируемый сетевой протокол. Протокол, объединяющий отдельные сети во всемирную сеть Интернет. Основа протокола – адресация компьютеров в сети (IP-адресация). Протокол работает на 3 (сетевом уровне), обеспечивая доставку пакетов данных между любыми узлами сети через произвольное число промежуточных узлов – маршрутизаторов.

Назначение протокола HTTP

HTTP (*HyperText Transfer Protocol* — **«протокол передачи гипертекста»**) — протокол прикладного уровня передачи данных (изначально в виде гипертекстовых документов). Основой HTTP является технология «клиент-сервер», то есть предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

HTTP используется также в качестве «транспорта» для других протоколов прикладного уровня, например **SOAP**.

Центральным объектом в HTTP является ресурс, на который указывает URI в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы.

Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д.

В отличие от многих других протоколов, HTTP является протоколом без памяти. Это означает, что протокол не хранит информацию о предыдущих запросах клиентов и ответах сервера.

Версии протокола HTTP

HTTP/0.9

Предложен в 1991г. как механизм для доступа к документам в Интернете и облегчения навигации посредством использования гипертекста. Упорядочены правила взаимодействия между клиентами и серверами HTTP и выделены функции между этими компонентами.

Задокументированы основные синтаксические и семантические положения.

HTTP/1.0

В 1996г. для практической реализации HTTP выпущен информационный документ RFC 1945, что послужило основой для реализации большинства компонентов HTTP/1.0.

HTTP/1.1

Принят в июне 1999 года. Новым в этой версии был режим «постоянного соединения»: TCP-соединение может оставаться открытым после отправки ответа на запрос, что позволяет посылать несколько запросов за одно соединение. Клиент обязан посылать информацию об имени хоста, к которому он обращается, что сделало возможной более простую организацию виртуального хостинга.

Структура сообщений протокола HTTP

"Классическая" схема HTTP-сеанса

Таким образом, после установки TCP-соединения клиент посылает серверу запрос, получает от него ответ, после чего взаимодействие прекращается. Обычно запрос клиента представляет собой требование передать HTML-документ или какой-нибудь другой ресурс, а ответ сервера содержит код этого ресурса.

Состав HTTP-сообщения

Запросы-ответы имеют форму HTTP-сообщений. Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

- Стартовая строка или строка состояния (Starting line) — определяет тип сообщения
- Заголовки (Headers) — характеризуют тело сообщения, параметры передачи и прочие сведения
- Тело сообщения (Message body) — непосредственно данные сообщения

Стартовая строка запроса

HTTP/0.9

GET URI

```
GET http://oak.oakland.edu/
```

HTTP/1.0 и выше

Метод URI HTTP/Версия протокола

```
GET http://oak.oakland.edu/ HTTP/1.1
```

- Метод — тип запроса. В версии HTTP 0.9 использовался только метод GET, список типов запросов для версии 1.1 составляет более 10 методов.
- URI определяет путь к запрашиваемому документу.

Стартовая строка ответа

HTTP/ Версия Код состояния Пояснение

```
HTTP/1.0 200 OK
```

- Версия — пара разделённых точкой арабских цифр как в запросе.
- Код состояния — три арабские цифры. По коду статуса определяется дальнейшее содержимое сообщения и поведение клиента.
- Пояснение — текстовое короткое пояснение к коду ответа для пользователя. Никак не влияет на сообщение и является необязательным, т.е. это строка символов, которая не обрабатывается клиентом. Она предназначена для системного администратора или оператора, занимающегося обслуживанием системы, и является расшифровкой кода ответа. В данном случае ответ означает успешное выполнение запроса

Метод запросов протокола HTTP

Метод HTTP — тип запроса, отправляемого веб-клиентом HTTP-серверу, указывающего на основную операцию над ресурсом. Задается последовательностью из любых символов кроме управляющих и разделителей (обычно метод представляет собой короткое английское слово записанное заглавными буквами). Название метода чувствительно к регистру. Если метод серверу неизвестен, отвечает ошибкой 501 (Method not implemented). Если серверу метод известен, но он не применим к конкретному ресурсу, то возвращается сообщение с кодом 405 (Method Not Allowed).

Наиболее часто используемые методы это **GET**, **HEAD** и **POST**.

Основные методы запросов протокола HTTP

Метод	Описание
GET	Используется для запроса содержимого ресурса с указанным URL. Получив запрос GET, сервер должен прочитать указанный ресурс и включить код ресурса в состав ответа клиенту. Ресурс, URL которого передается в составе запроса, не обязательно должен представлять собой HTML-страницу, файл с изображением или другие данные. URL ресурса может указывать на исполняемый код программы, который, при соблюдении определенных условий, должен быть запущен на сервере. Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа «?»
HEAD	Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения. Метод HEAD является модификацией метода GET.

Основные методы запросов протокола HTTP

Метод	Описание
POST	Применяется для передачи пользовательские данных заданному ресурсу. При этом передаваемые данные (в примере с блогами — текст комментария) включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы. Не смотря на то, что основное назначение метода POST - передача данных на сервер, метод POST нередко используется для получения информации с сервера. Как и в случае с методом GET, URL, заданный в строке состояния, указывает на конкретный ресурс. Метод POST также может использоваться для запуска процесса.
PUT	Применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существовало ресурса, то сервер создаёт его и возвращает статус 201 (Created). Если же был изменён ресурс, то сервер возвращает 200 (Ok) или 204 (No Content). Метод PUT является модификацией метода POST.

Основные методы запросов протокола HTTP

Метод	Описание
PATCH	Аналогично PUT, но применяется только к фрагменту ресурса.
DELETE	Удаляет указанный ресурс
TRACE	Возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавляют или изменяют в запросе
LINK	Устанавливает связь указанного ресурса с другими
UNLINK	Удаляет связь указанного ресурса с другими

Метод запросов протокола HTTP

При выборе метода (GET\POST) следует учитывать следующие факторы:

- метод GET ограничивает объем передаваемой информации;
- метод GET открыто пересылает введенную информацию в обрабатывающий сценарий, что может неблагоприятно сказаться на безопасности. Например, каждый человек, которому виден монитор компьютера, может заметить введенный в форму пароль;
- страницу, сгенерированную формой с помощью метода GET, разрешается помечать закладкой, а сгенерированную методом POST нет.

Заголовки запросов протокола HTTP

Заголовки HTTP — это строки в HTTP-сообщении, содержащие разделённую двоеточием пару параметр-значение. Формат заголовков соответствует общему формату заголовков текстовых сетевых сообщений. Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой.

Примеры заголовков:

```
Server: Apache/2.2.11 (Win32) PHP/5.3.0
Last-Modified: Sat, 16 Jan 2010 21:16:42 GMT
Content-Type: text/plain; charset=windows-1251
Content-Language: ru
```

Каждая строка представляет собой один заголовок. При этом то, что находится до первого двоеточия, называется именем, а что после него — значением.

Поля заголовка запроса HTTP

Поле	Значение
Host	Доменное имя или IP-адрес узла, к которому обращается клиент
Referer	URL документа, который ссылается на ресурс, указанный в строке состояния
From	Адрес электронной почты пользователя, работающего с клиентом
Accept	MIME-типы данных, обрабатываемых клиентом. Это поле может иметь несколько значений, отделяемых одно от другого запятыми. Часто поле заголовка Accept используется для того, чтобы сообщить серверу о том, какие типы графических файлов поддерживает клиент
Accept-Language	Набор двухсимвольных идентификаторов, разделенных запятыми, которые обозначают языки, поддерживаемые клиентом
Accept-Charset	Перечень поддерживаемых наборов символов
Content-Type	MIME-тип данных, содержащихся в теле запроса (если запрос не состоит из одного заголовка)
Content-Length	Число символов, содержащихся в теле запроса (если запрос не состоит из одного заголовка)
Range	Присутствует в том случае, если клиент запрашивает не весь документ, а лишь его часть
Connection	Используется для управления TCP-соединением. Если в поле содержится Close, это означает, что после обработки запроса сервер должен закрыть соединение. Значение Keep-Alive предлагает не закрывать TCP-соединение, чтобы оно могло быть использовано для последующих запросов
User-Agent	Информация о клиенте

Пример HTML-запроса, сгенерированного браузером

```
http://ocsp.verisign.com/

POST / HTTP/1.1
Host: ocsp.verisign.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:17.0)
Gecko/20100101 Firefox/17.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Content-Length: 115
Content-Type: application/ocsp-request
```

Поля заголовка ответа HTTP

Имя поля	Описание содержимого
Server	Имя и номер версии сервера
Age	Время в секундах, прошедшее с момента создания ресурса
Allow	Список методов, допустимых для данного ресурса
Content-Language	Языки, которые должен поддерживать клиент для того, чтобы корректно отобразить передаваемый ресурс
Content-Type	MIME-тип данных, содержащихся в теле ответа сервера
Content-Length	Число символов, содержащихся в теле ответа сервера
Last-Modified	Дата и время последнего изменения ресурса
Date	Дата и время, определяющие момент генерации ответа
Expires	Дата и время, определяющие момент, после которого информация, переданная клиенту, считается устаревшей
Location	В этом поле указывается реальное расположение ресурса. Оно используется для перенаправления запроса
Cache-Control	Директивы управления кэшированием. Например, no-cache означает, что данные не должны кэшироваться

Пример HTML-ответа сервера на запрос, представленный выше

```
HTTP/1.0 200 Ok
Last-Modified: Thu, 10 Jan 2013 20:43:36 GMT
Expires: Thu, 17 Jan 2013 20:43:36 GMT
Content-Type: application/ocsp-response
Content-Transfer-Encoding: binary
Content-Length: 1987
Cache-Control: max-age=589347, public, no-transform, must-
revalidate
Date: Fri, 11 Jan 2013 01:01:09 GMT
Connection: Keep-Alive
```

MIME –типы данных

Поле с именем Content-type может встречаться как в запросе клиента, так и в ответе сервера. В качестве значения этого поля указывается MIME –тип содержимого запроса или ответа. MIME –тип также передается в поле заголовка Accept, присутствующего в запросе.

Спецификация MIME (Multipurpose Internet Mail Extension — многоцелевое почтовое расширение Internet) первоначально была разработана для того, чтобы обеспечить передачу различных форматов данных в составе электронных писем. Однако применение MIME не исчерпывается электронной почтой. Средства MIME успешно используются в www и, по сути, стали неотъемлемой частью этой системы.

В соответствии со спецификацией MIME, для описания формата данных используются тип и подтип. Тип определяет, к какому классу относится формат содержимого HTTP-запроса или HTTP-ответа. Подтип уточняет формат. Тип и подтип отделяются друг от друга косой чертой: тип/подтип

Основные MIME-типы данных

Тип/подтип	Расширение файла	Описание
application/pdf	.pdf	Документ, предназначенный для обработки Acrobat Reader
application/msexcel	.xls	Документ в формате Microsoft Excel
application/postscript	.ps, .eps	Документ в формате PostScript
application/x-tex	.tex	Документ в формате TeX
application/msword	.doc	Документ в формате Microsoft Word
application/rtf	.rtf	Документ в формате RTF, отображаемый с помощью Microsoft Word
image/gif	.gif	Изображение в формате GIF
image/ jpeg	.jpeg, .jpg,	Изображение в формате JPEG
image/tiff	.tiff, .tif	Изображение в формате TIFF
image/x-xbitmap	.xbm	Изображение в формате XBitmap
text/plain	.txt	ASCII-текст
text/html	.html , .htm	Документ в формате HTML
audio/midi	.midi, .mid	Аудиофайл в формате MIDI
audio/x-wav	.wav	Аудиофайл в формате WAV
message/rfc822		Почтовое сообщение
message/news		Сообщение в группы новостей
video /mpeg	.mpeg, .mpg, .mpe	Видеофрагмент в формате MPEG
video/avi	.avi	Видеофрагмент в формате AVI

Коды состояния протокола HTTP

Код состояния HTTP (*HTTP status code*) — часть первой строки ответа сервера при запросах по протоколу HTTP. Он представляет собой целое число из трех арабских цифр. Первая цифра указывает на *класс состояния*. За кодом ответа обычно следует отделённая пробелом поясняющая фраза на английском языке, которая разъясняет человеку причину именно такого ответа. Примеры:

- `201 Webpage Created`
- `403 Access allowed only for registered users`
- `507 Insufficient Storage`

В настоящее время выделено **пять** классов кодов состояния.

1. 1xx Informational (Информационный)

1. *100 Continue* — сервер удовлетворён начальными сведениями о запросе, клиент может продолжать пересылать заголовки. Появился в HTTP/1.1.

2. *101 Switching Protocols* — сервер предлагает перейти на более подходящий для указанного ресурса протокол; список предлагаемых протоколов сервер обязательно указывает в поле заголовка Update. Если клиента это заинтересует, то он посылает новый запрос с указанием другого протокола. Появился в HTTP/1.1.

3. *102 Processing* — запрос принят, но на его обработку понадобится длительное время. Используется сервером, чтобы клиент не разорвал соединение из-за превышения времени ожидания. Клиент при получении такого ответа должен сбросить таймер и дожидаться следующей команды в обычном режиме.

2. 2xx Success (Успех)

1. *200 OK* — успешный запрос. Если клиентом были запрошены какие-либо данные, то они находятся в заголовке и/или теле сообщения. Появился в HTTP/1.0.
2. *201 Created* — в результате успешного выполнения запроса был создан новый ресурс. Сервер должен указать его местоположение в заголовке Location. Серверу рекомендуется ещё указывать в заголовке характеристики созданного ресурса (например, в поле Content-Type). Если сервер не уверен, что ресурс действительно будет существовать к моменту получения данного сообщения клиентом, то лучше использовать ответ с кодом 202. Появился в HTTP/1.0.
3. *202 Accepted* — запрос был принят на обработку, но она не завершена. Клиенту не обязательно дожидаться окончательной передачи сообщения, так как может быть начат очень долгий процесс. Появился в HTTP/1.0.
4. *204 No Content* — сервер успешно обработал запрос, но в ответе были переданы только заголовки без тела сообщения. Клиент не должен обновлять содержимое документа, но может применить к нему полученные метаданные. Появился в HTTP/1.0.

3. 3xx Redirection (Перенаправление)

1. **301 Moved Permanently** — запрошенный документ был окончательно перенесен на новый URI, указанный в поле Location заголовка. Некоторые клиенты некорректно ведут себя при обработке данного кода. Появился в HTTP/1.0.

2. **302 Found, 302 Moved Temporarily** — запрошенный документ временно доступен по другому URI, указанному в заголовке в поле Location. Введено в HTTP/1.0.

3. **304 Not Modified** — сервер возвращает такой код, если клиент запросил документ методом GET, использовал заголовок If-Modified-Since или If-None-Match и документ не изменился с указанного момента. При этом сообщение сервера не должно содержать тела. Появился в HTTP/1.0.

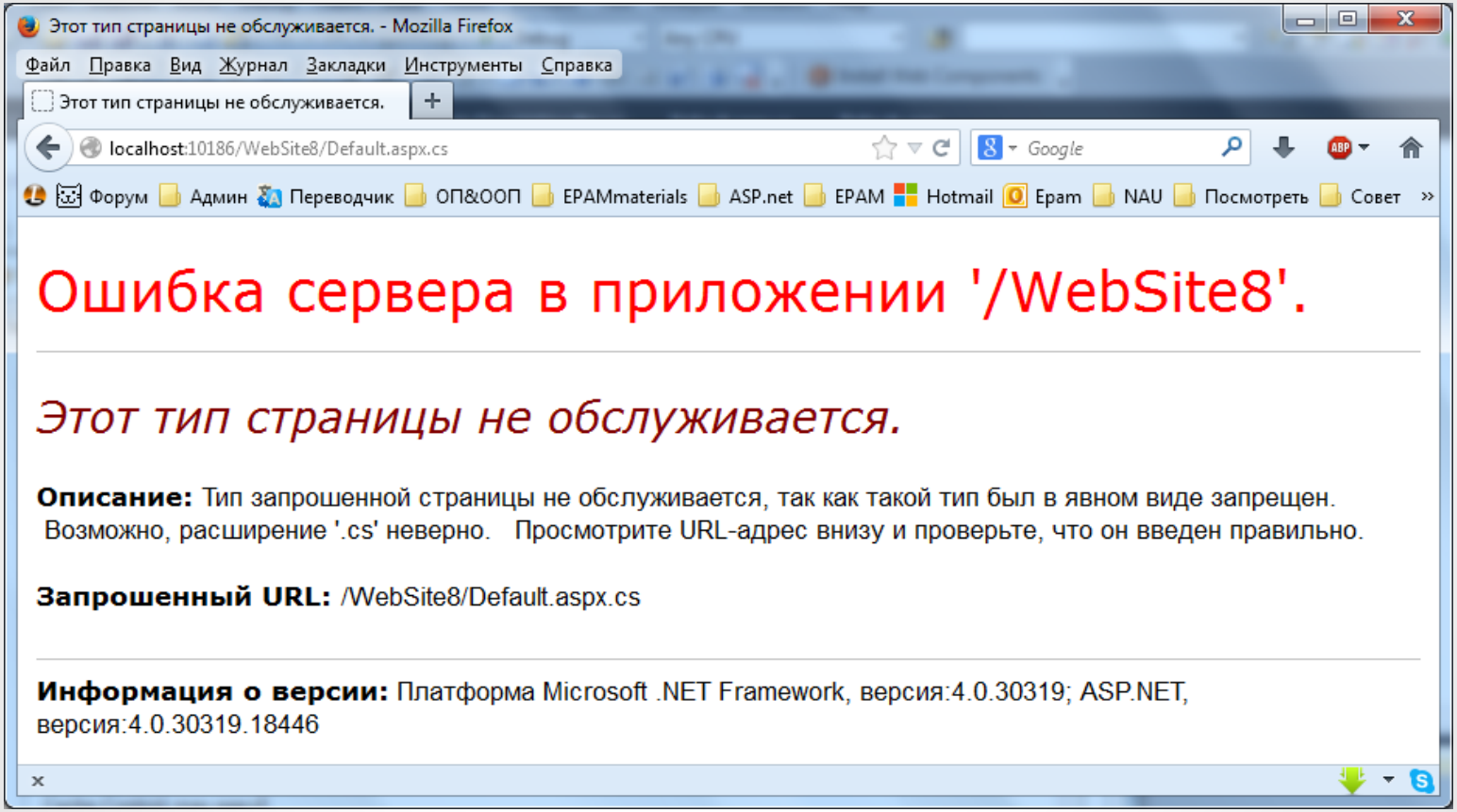
4. 4xx Client Error (Ошибка клиента)

1. **400 Bad Request** — сервер обнаружил в запросе клиента синтаксическую ошибку. Появился в HTTP/1.0..
2. **401 Unauthorized** — запрос требует идентификации пользователя. Сервер должен запросить имя и пароль у пользователя, а тот передаст их в заголовке WWW-Authenticate в следующем запросе. Если были указаны неверные данные, то сервер снова вернёт этот же статус. Появился в HTTP/1.0.
3. **402 Payment Required** — предполагается использовать в будущем. В настоящий момент не используется. Этот код предусмотрен для платных пользовательских сервисов, а не для хостинговых компаний. Имеется в виду, что эта ошибка не будет выдана хостинговым провайдером в случае просроченной оплаты его услуг. Зарезервирован, начиная с HTTP/1.1.
4. **403 Forbidden** — ответ на запрос ресурса, доступ к которому запрещен. Сервер понимает запрос, но отказывается его выполнять из-за ограничений в доступе для клиента к указанному ресурсу. Если для доступа к ресурсу требуется аутентификация средствами HTTP, то сервер вернёт ответ 401 или 407 при использовании прокси. В противном случае ограничения были заданы администратором сервера или разработчиком веб-приложения и могут быть любыми в зависимости от возможностей используемого ПО. Появился в HTTP/1.0.

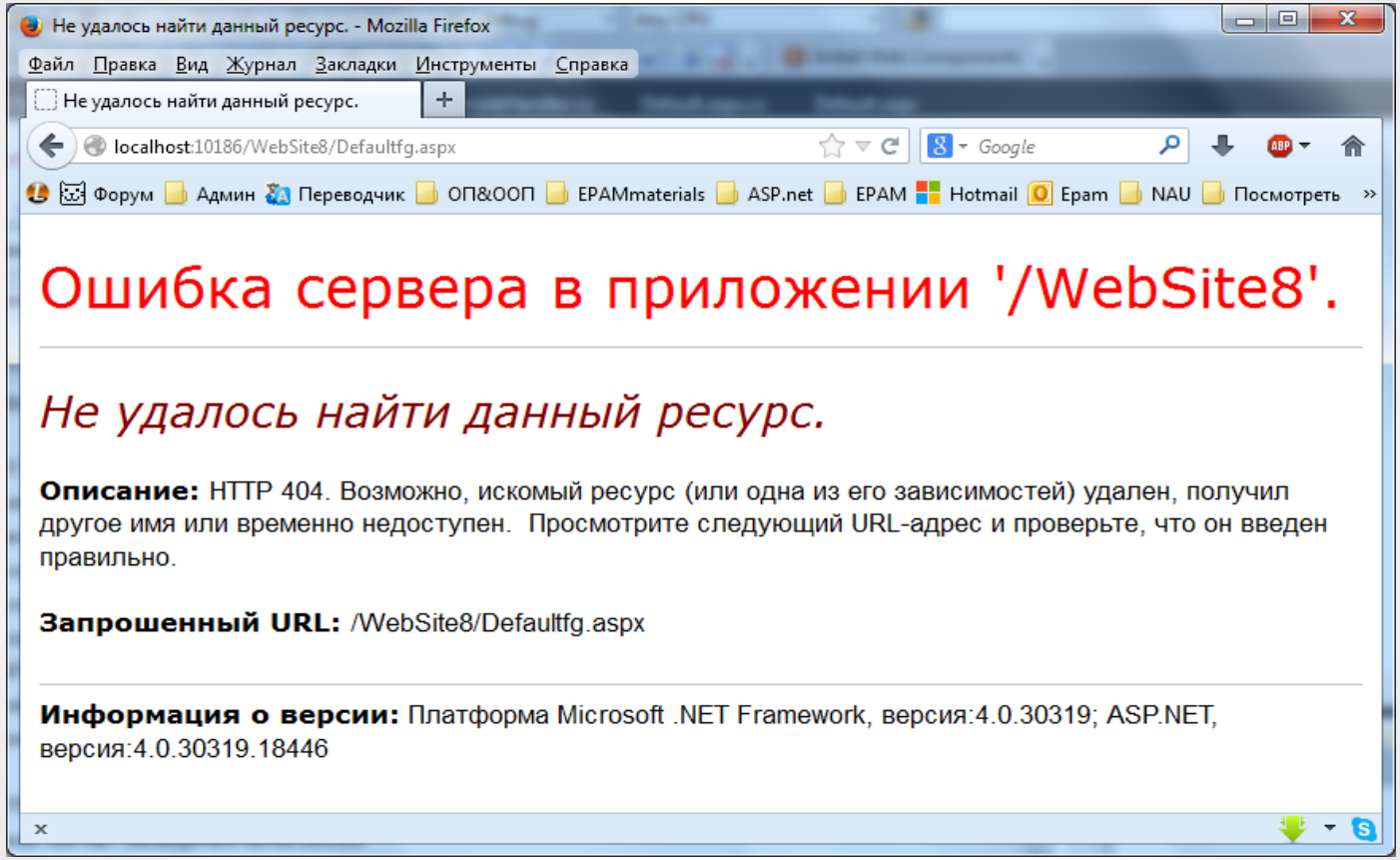
4. 4xx Client Error (Ошибка клиента)

5. **404 Not Found** — самая распространенная ошибка при пользовании Интернетом, основная причина — ошибка в написании адреса Web-страницы. Сервер понял запрос, но не нашёл соответствующего ресурса по указанному URI. Ответ 404 может использоваться вместо 403, если требуется тщательно скрыть от посторонних глаз определённые ресурсы. Появился в HTTP/1.0.
6. **405 Method Not Allowed** — указанный клиентом метод нельзя применить к текущему ресурсу. В ответе сервер должен указать доступные методы в заголовке Allow, разделив их запятой. Эту ошибку сервер должен возвращать, если метод ему известен, но он не применим именно к указанному в запросе ресурсу, если же указанный метод не применим на всём сервере, то клиенту нужно вернуть код 501. Появился в HTTP/1.1.
7. **408 Request Timeout** — время ожидания сервером передачи от клиента истекло. Клиент может повторить аналогичный предыдущему запрос в любое время. Например, такая ситуация может возникнуть при загрузке на сервер объёмного файла методом POST или PUT.

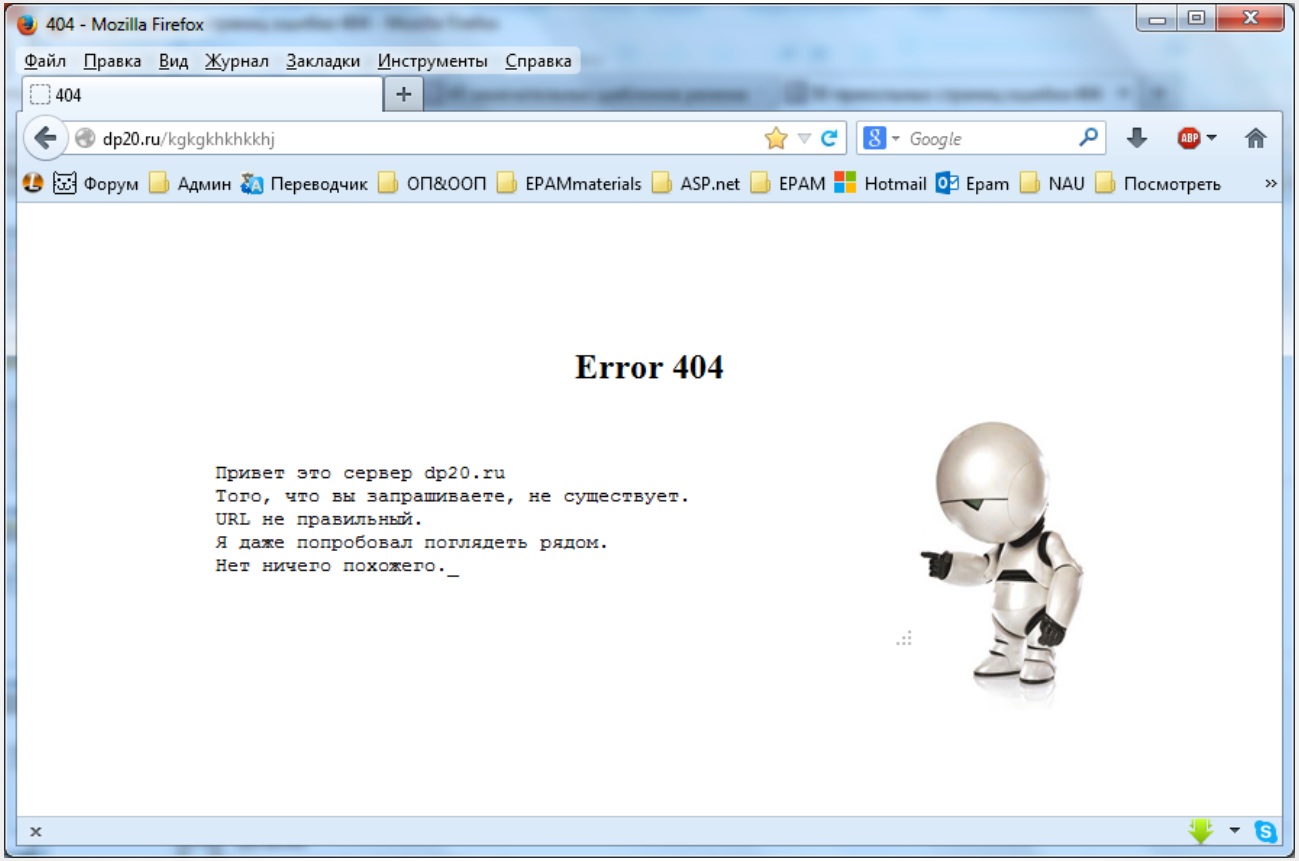
Ошибка клиента 403 – forbidden



Ошибка клиента 404 – Not Found

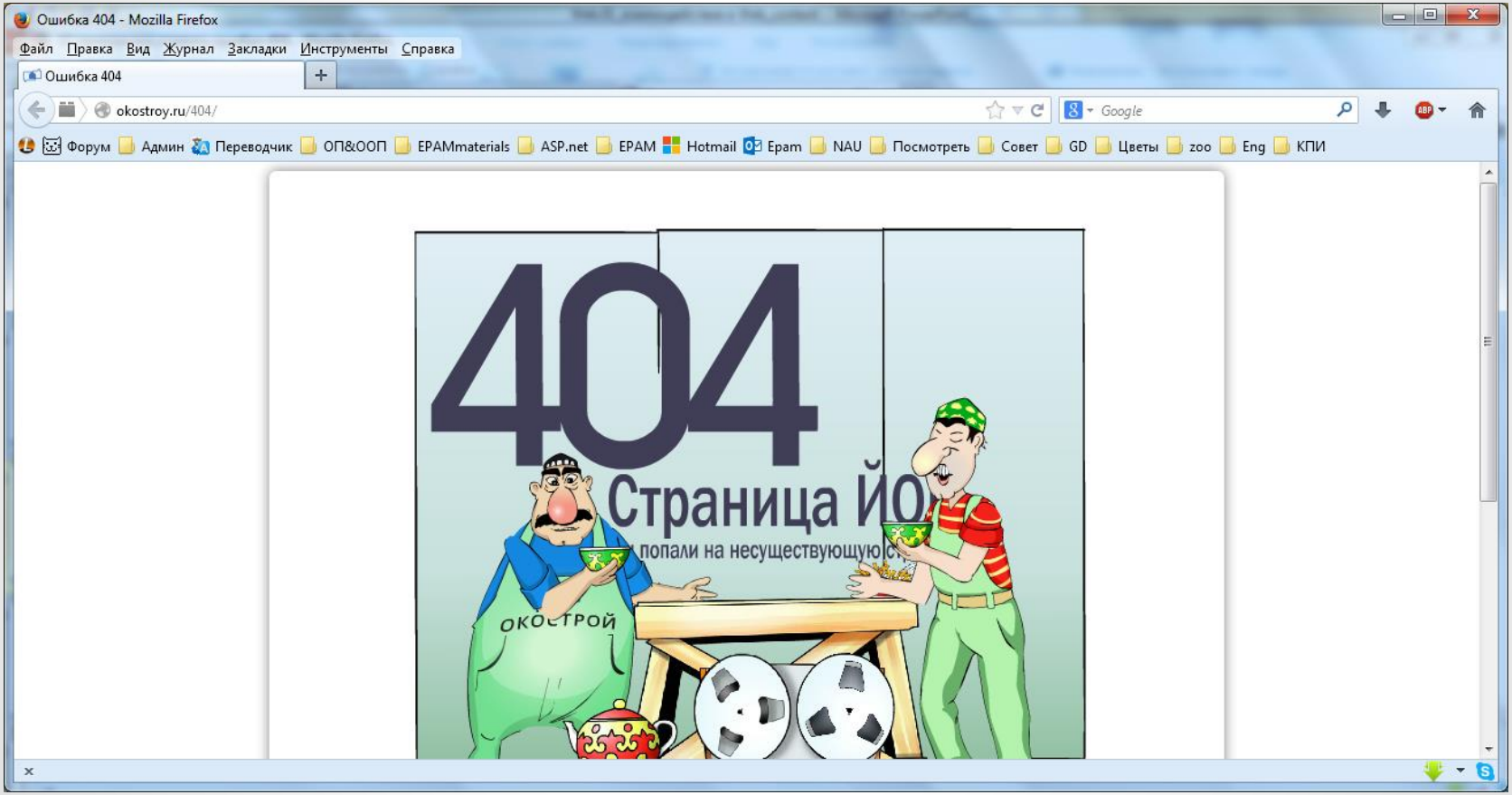


Ошибка клиента 404 – Not Found

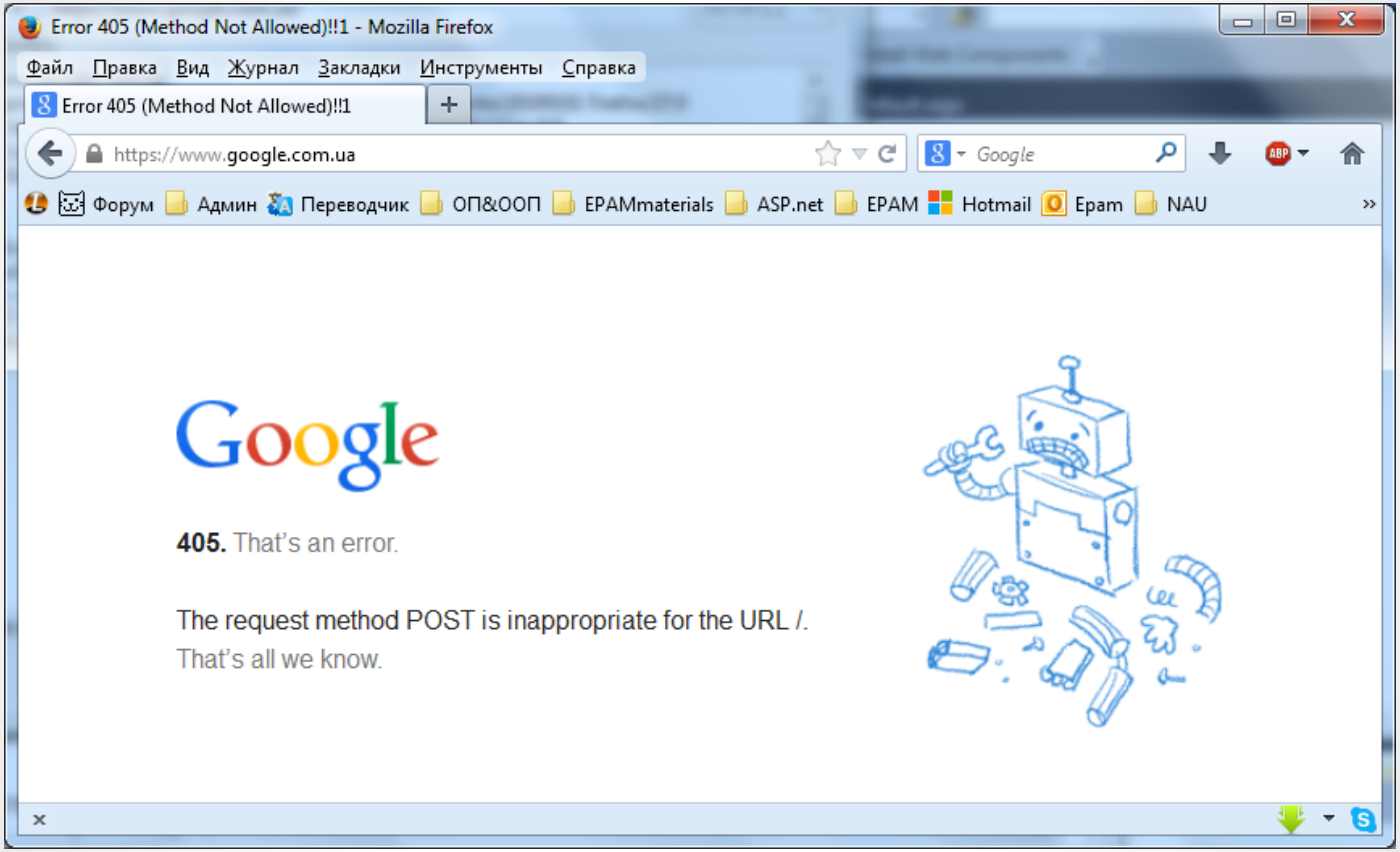


Ошибка клиента 404 – Not Found

okostroy.ru/vayvayvaaaaay – запросить ресурс ☺



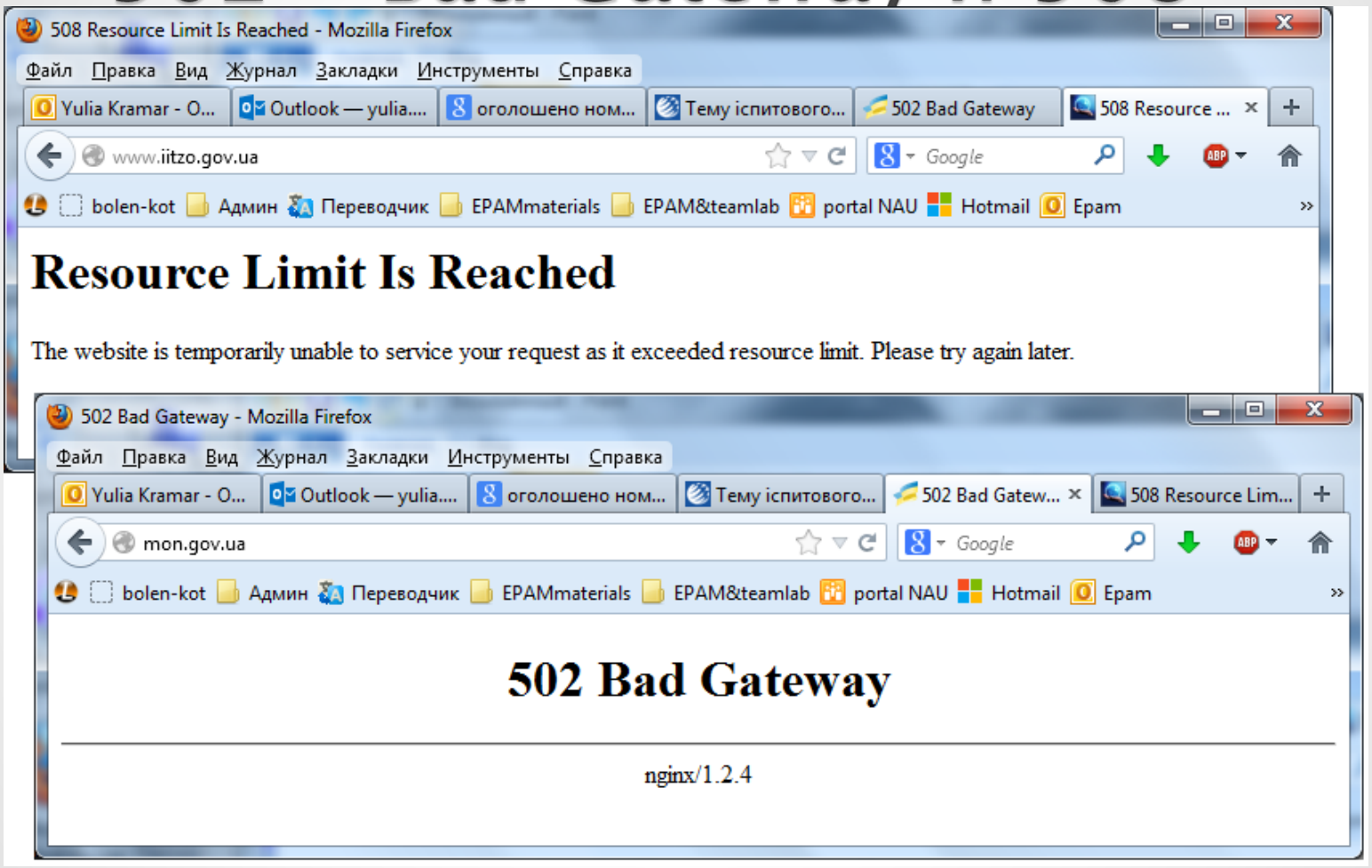
Ошибка клиента 405 – Method Not Allowed



5. 5xx Server Error (Ошибка сервера)

1. **500 Internal Server Error** — любая внутренняя ошибка сервера, которая не входит в рамки остальных ошибок класса. Появился в HTTP/1.0.
2. **501 Not Implemented** — сервер не поддерживает возможностей, необходимых для обработки запроса. Типичный ответ для случаев, когда сервер не понимает указанный в запросе метод. Если же метод серверу известен, но он не применим к данному ресурсу, то нужно вернуть ответ 405. Появился в HTTP/1.0.
3. **503 Service Unavailable** — сервер временно не имеет возможности обрабатывать запросы по техническим причинам (обслуживание, перегрузка и прочее). В поле Retry-After заголовка сервер может указать время, через которое клиенту рекомендуется повторить запрос. Появился в HTTP/1.0.
4. **505 HTTP Version Not Supported** — сервер не поддерживает или отказывается поддерживать указанную в запросе версию протокола HTTP. Появился в HTTP/1.1.

Ошибки сервера
502 – Bad Gateway и 508 –



Web. Asp .Net

Ресурсы: