## Архитектура клиент-сервер

**Клиент-сервер** — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисами), называемыми серверами, и заказчиками услуг, называемыми клиентами. Нередко клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением.

Частным случаем клиент серверной архитектуры является **трехзвенная архитектура.**

В зависимости от назначения и способа взаимодействия с информацией в большинстве систем можно выделить три основных компонента:

- визуализация (В)

- обработка данных (О)

- хранение данных (Д)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Клиент | Data Server | Классический  сервер | Тонкий клиент |
|  |  |  |  |  |
| Клиент | В |  |  |  |
|  | О | В |  |  |
|  | Д | О | В |  |
|  |  | Д | О | В |
|  |  |  | Д | О |
| Сервер |  |  |  | Д |
|  |  |  |  |  |
|  |  |  |  |  |

От того как распределены перечисленные компоненты между клиентом и сервером можно судить об архитектуре системы.

Случай первый. Все три компонента находятся на стороне клиента. Хранение информации, ее обработка и отображение осуществляются в пределах одного компьютера. Пример – обычный пользовательский компьютер.

Случай второй. Data Server (сервер – хранилище данных).

Случай третий. Классический сервер. Обработка и хранение данных осуществляются на стороне сервера. Подготовленные специальным образом передаются клиенту. Клиент визуализирует полученную информацию. Так работает Интернет. В качестве клиента выступает браузер.

Случай четвертый. Так называемый тонкий клиент. Хранение, обработка и визуализация данных осуществляются на стороне сервера. В качестве примера можно привести – терминал. Один сервер и несколько терминалов (монитор с клавиатурой).

## Протокол HTTP

Протокол – набор правил, определяющий способы взаимодействия между компонентами.

Начнем с того что ваш браузер (когда вы набрали **URL**) соединяется по протоколу **HTTP** с указанным сервером и просит у него нужный файл, примерно так:

**GET /cgi-bin/hello.cgi HTTP/1.1**

**Метод** **URI** HTTP/**Версия**

Вот это самое главное в запросе.

Здесь:

**Метод** - название запроса, одно слово заглавными буквами. В качестве метода может быть GET, POST, HEAD, PUT, DELETE.

[**URI**](http://ru.wikipedia.org/wiki/URI) - определяет путь к запрашиваемому документу.

**Версия** — пара разделённых точкой арабских цифр. Например: 1.1.

Далее браузер посылает различную информацию о себе. (К примеру, **Accept:** \*/\*).

Если существующий файл найден, то сервер отошлет браузеру ответ:

**HTTP/1.0 200 Okay**

**Content-Type: text/html**

**<HTML>**

**<BODY>**

**<H1>Hello world!</H1>**

**...**

**</BODY>**

**</HTML>**

В ответе, состоящем из заголовка и тела, в заголовке содержится код возврата и информация о типе содержимого. Далее после пустой строки (она нужна чтоб отделить **заголовок** от **тела**) идет информация из самого документа , по заданому **URL** <HTML><BODY>...

## CGI

Рассмотрим подробнее, что происходит на стороне сервера при обращении к файлу hello.cgi.

Расширение файла ‘.cgi’ говорит нам о том, что это выполняемое CGI приложение. CGI – Common Gateway Interface (общий интерфейс шлюза). Это стандарт интерфейса, используемого для связи внешней программы с веб-сервером. (Wikipedia).

Информация, которую посылает сервер по своей природе статична. Сервер не может вам послать больше того что у него есть на момент вашего запроса. Вот чтобы изменить эту ситуацию к лучшему и был введен стандарт CGI. CGI – расширяет возможности Web-сервера, тем что информация поступаемая от Web-сервера приобретает динамический характер. Делается это следующим образом: Если в URL (в броузере) указать не какой-нибудь статический ресурс, а специальную программу (CGI-скрипт) . А сервер проанализировав такой запрос возьмет и запустит ее. А она в свою очередь (пользуясь всеми ресурсами сервера) выдаст некоторую динамическую информацию (например: страничку) . А Web-сервер после окончания работы CGI-скрипта отправит эту информацию броузеру предварительно снабдив ее нужным для протокола [HTTP](http://students.uni-vologda.ac.ru/pages/pm97/cgi/http.html#HRes) заголовком.

Так вот CGI –является стандартом, который описывает как сервер должен запускать CGI-скрипт, как должен передавать ему параметры HTTP-запроса и как CGI-скрипт должен передавать результаты своей работы серверу.

CGI программа может быть написана на любом языке программирования.

Пример на Pascal:

PROGRAM PrintHello(INPUT, OUTPUT);

BEGIN

WRITELN(‘Content-Type: text/plain’);

WRITELN;

WRITELN(‘Hello world!’);

END.

А вот такой же пример на языке Perl:

**#!/usr/bin/perl**

**#first.cgi**

**print "Content-Type: text/html\n\n";**

**print "<HTML><BODY>";**

**print "<H1>Hello world!!!</H1>";**

**print "</BODY></HTML>";**

Обратите внимание на первую строку программы «#!/usr/bin/perl». В ней указан путь до интерпретатора perl. У нас путь будет «#!perl».

Обратите внимание на отделение заголовка от тела.

Грубая ошибка с логической точки зрения это пропустить вывод пустой строки, которая отделяет заголовок от тела:

**print "Content-Type: text/html\n\n"; #Все Правильно**

**print "Content-Type: text/html\n"; #ОШИБКА!!!**

## Параметры окружения

В приведенных примерах нет ничего особенно замечательного. Скрипт просто вываливает HTML’ый текст, который может быть отображен в браузере. По-настоящему огромную мощь придает **CGI** возможность обработки параметров, которые переданы скрипту. Например, вы можете набрать  
**http://www.somehost.ru/ cgi-bin/greeting.cgi?name =Sarah**

С точки зрения пользователя, вы ожидаете, что скрипт выведет приветственное сообщение на основе данных переданных в теле запроса. К примеру так ‘Hello, Sarah!’

Это с точки зрения пользователя... А на сервере при запуске **CGI**-скрипта сервер формирует среду окружения в которой скрипт может найти всю доступную информацию о **HTTP**-соединении и о запросе.

REQUEST\_METHOD

Это одно из самых главных поле используемое для определения метода запроса HTTP Протокол HTTP использует методы GET и POST для запроса к серверу.Они отличаются тем что при методе GET запрос является как-бы частью URL т.е. http://www..../myscript.cgi?request а при методе POST данные передаются в теле HTTP-запроса (при GET тело запроса пусто) и следовательно для CGI тоже есть различие при GET запрос идет в переменную QUERY\_STRING а при POST подается на STDIN скрипта.  
Пример:REQUEST\_METHOD=GET

QUERY\_STRING

Это строка запроса при методе GET. Вам всем известно что запрос из формы кодируется браузером поскольку не все символы разрешены в URL некоторые имеют специальное назначение. Теперь о методе urlencode: неплохо бы чисто формально напомнить, что все пробелы заменяются в URL на знак '+', а все специальные и непечатные символы на последовательность %hh ,где hh-шестнадцатеричный код символа, разделитель полей формы знак '&',так что при обработке форм надо произвести декодирование.  
Пример:QUERY\_STRING= name=quake+doomer&age=20&hobby=games

CONTENT\_LENGTH

Длина в байтах тела запроса. При методе запроса POST необходимо считать со стандартного входа STDIN CONTENT\_LENGTH байт, а потом производить их обработку. Обычно методом POST пользуются для передачи форм, содержащих потенциально большие области ввода текста TEXTAREA. При этом методе нет никаких ограничений, а при методе GET существуют ограничения на длину URL .  
Пример:CONTENT\_LENGTH=31

CONTENT\_TYPE

Тип тела запроса(для форм кодированных выше указанным образом он application/x-www-form-urlencoded)

GATEWAY\_INTERFACE

Версия протокола CGI.  
Пример:GATEWAY\_INTERFACE=CGI/1.1

REMOTE\_ADDR

IP-Адрес удаленного хоста, делающего данный запрос.  
Пример:REMOTE\_ADDR=139.142.24.157

REMOTE\_HOST

Если запрашивающий хост имеет доменное имя, то эта переменная содержит его, в противном случае -тот же самый IP-адрес что и REMOTE\_ADDR  
Пример:REMOTE\_HOST=idsoftware.com

SCRIPT\_NAME

Имя скрипта,исполизованое в запросе.Для получения реального пути на сервере используйте SCRIPT\_FILENAME  
Пример:SCRIPT\_NAME=/~paaa/guestbook.cgi

SCRIPT\_FILENAME

Имя файла скрипта на сервере.  
Пример:SCRIPT\_FILENAME=/home/p/paaa/public\_html/cgi-bin/guestbook.cgi

SERVER\_NAME

Имя серера ,чаще всего доменное как www.microsoft.com ,но в редких случаях за неимением такового может быть IP-адресом как 157.151.74.254  
Пример:SERVER\_NAME=www.uic.nnov.ru

SERVER\_PORT

TCP-Порт сервера используюшийся для соединения .По умолчаниию HTTP-порт 80, хотя может быть в некоторых случаях другим.  
Пример:SERVER\_PORT=80

SERVER\_PROTOCOL

Версия протокола сервера.  
Пример:SERVER\_PROTOCOL=HTTP/1.1

SERVER\_SOFTWARE

Програмное обеспечение сервера.  
Пример:Apache/1.0

AUTH\_TYPE, REMOTE\_USER

Эти переменные определены в том случае,когда запрошеный ресурс требует аутентификации пользователя.

Переменные заголовка HTTP-запроса.  
За исключением тех строк из заголовка HTTP-запроса которые были включены в другие переменные,сервер приделывает строкам префикс HTTP\_ и заменяет знаки '-' на '\_':  
HTTP\_ACCEPT

Давая запрос на сервер браузер обычно расчитывает получить информацию определеного формата,и для этого он в заголовке запроса указывает поле Accept:,Отсюда скрипту поступает cписок тех MIME,которые браузер готов принять в качестве ответа от сервера.  
Пример:HTTP\_ACCEPT=text/html,text/plain,image/gif

HTTP\_USER\_AGENT

Браузер обычно посылает на сервер и информацию о себе,чтоб базируясь на знании особеностей и недостатков конкретных браузеров CGI-скрипт мог выдать информацию с учетом этого. Например,разные браузеры могут поддерживать или не поддерживать какие-то HTMLые тэги.  
Пример:HTTP\_USER\_AGENT=Mozila/2.01 Gold(Win95;I)

HTTP\_HOST

Имя хоста к которому обращается браузер. Так как физически на одном сервере может находиться сразу много серверов (Виртуальные Хосты), то должен быть способ сообщить серверу к какому именно идет обращение. Скрипт же может тоже в зависимости от этой переменной производить различные действия, таким если он используется на сайтах сразу нескольких виртуальных хостов.  
Пример:HTTP\_HOST=www.nnov.city.ru

## Передача параметров методом GET

Доступ к переменным окружения осуществляется в скриптах на языке Perl также очень просто. Всякий раз при запуске скрипта Perl помещает копии переменных окружения в ассоциативный массив с именем % ENV .

**#!perl**

**#vars.cgi**

**sub urldecode{ #очень полезная функция декодирования**

**local($val)=@\_; #запроса**

**$val=~s/\+/ /g;**

**$val=~s/%([0-9A-H]{2})/pack('C',hex($1))/ge;**

**return $val;**

**}**

**print "Content-Type: text/html\n\n";**

**print "<HTML>\n";**

**print "<BODY>\n";**

**print "REQUEST\_METHOD = $ENV{'REQUEST\_METHOD'}\n";**

**print "Все переменные окружения:<BR>\n";**

**foreach $env\_var (keys %ENV){**

**print "<I>$env\_var=$ENV{$env\_var}</I><BR>\n";**

**}**

**print "</BODY></HTML>\n";**

В конце программы вы можете увидеть цикл, который выводит на экран все переменные окружения. Обязательно посмотрите результат.

## Заголовки запросов и ответов

Этапы соединения  
Первый этап это когда **HTTP** –клиент (браузер) соединяется с сервером. Для этого он использует протокол **TCP/IP** соединение происходит к известному клиенту **TCP**-порту (**80** -номер порта **HTTP**) (другие сервисы сидят на других портах ,например **FTP** и **SMTP** на **21** и **25**)  
Вторым этапом идет запрос клиента: клиент передает заголовок запроса и возможно(в зависимости от метода) тело сообщения запроса.В заголовке обязательно указывается метод ,**URI**,и версия **HTTP**,и может быть еще несколько необязательных полей  
Третий этап -ответ сервера,который опять таки состоит из заголовка,в котором сервер указывает версию HTTP и код статуса, который может говорить о успешном или неуспешном результате и его причинах.Далее идет тело ответа.  
Четвертым этапом происходит разрыв **TCP/IP** соединения.

HTTP -запрос  
Запрос состоит из Строки запроса(она обязательна) и остальных полей. Синтаксис строки :**МЕТОД <SP> URI <SP> HTTP/версия <CRLF>**  
где <SP> -пробел ,<CRLF> -переход на новую строку  
***Методы HTTP.***  
**GET**  
Самый часто применяемый метод,в протоколе **HTTP/0.9** был единственным методом,и применяется для извлечения информации по заданому **URI** Может быть условным если в заголовке указано поле **If-Modified-Since:**  
  
**HEAD**  
Почти идентичен **GET** но отличается тем что сервер не возвращает тело обьекта а только его заголовок (метаинформацию) программы могут применять его для проверки гиперссылок на правильность,доступность и изменения.  
  
**POST**  
передает данные для обработки их программой ,указаной в **URI** сдесь обязательно указывается поле **Content-Length:**  
  
Сушествуют и другие ,реже применяемые методы,например **PUT** -для сохранения передавемых данных в указаном **URI** и **DELETE** для удаления ресурса.  
  
Поля заголовка запроса  
После строки запроса идут поля заголовка запроса. Поля общего(general-header) заголовка (он общий как для запросов так и для ответов):  
**Date:**  
Указывает дату запроса,например:  
**Date: Sun, 20 Nov 1994 08:12:31 GMT**  
  
**MIME-version:**  
Указывает версию **MIME** (по умолчанию **1.0**)  
**MIME-version: 1.0**  
  
**Pragma:**  
Содержит указания для таких промежуточных агентов как прокси и шлюзы,  
**Pragma: no-cache**  
  
Поля относящиеся к запросу(**Request-Header**):  
**Authorization:**  
Содержит информацию аутентификации  
**Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==**  
  
**From:**  
Браузер может посылать адрес пользователя серверу  
**From: quake@doom.ru**  
  
**If-Modified-Since:**  
используется при методе **GET** ресурс возвращается ,если он был изменен с указаного момента, может использоваться при кешировании.  
**If-Modified-Since:Mon 15 Jul 1997 00:15:24 GMT**  
  
**Referer:**  
Содержит **URL** предшествующего ресурса.  
**Referer: http://www.uic.nnov.ru/~paaa/index.html**  
  
**User-Agent:**  
Програмное обеспечение клиента.  
**User-Agent: Mozilla/3.0**  
  
Заголовок информации сообщения (**Entity-Header**) применяется как в запросах так и в ответах (при этом некоторые поля только в ответах):  
**Allow:** (в ответе сервера)  
Список методов,поддерживаемых ресурсом.  
**Allow: GET, HEAD**  
  
**Content-Encoding:**  
идентифицирует метод кодировки,которым был закодирован ресурс  
**Content-Encoding: x-gzip**  
  
**Content-Length:**  
Длина тела сообщения  
**Content-Length: 102**  
  
**Content-Type:**  
Содержит тип ресурса(**MIME**),для текстовых еще и кодировку символов(необязательно)  
**Content-Type: text/html; charset=windows-1251**  
  
**Expires:** (в ответе сервера)  
Дата окончания действия ресурса,применяется в кешировании для запрета кеширования устаревших ресурсов (в ответе)  
**Expires: Tue, 24 Sep 1998 23:00:15 GMT**  
  
**Last-Modified:** (в ответе сервера)  
Время последнего обновления ресурса  
**Last-Modified: Tue, 23 sep 1998 13:48:40 GMT**  
  
**Другие поля:**  
Поля **Accept:** указывают серверу выдавать только указаные форматы данных,которые клиент может распознать.  
**Accept: text/html**  
**Accept: text/plain**  
**Accept: image/gif**  
  
Поле **Host:** служит для того , чтобы указать, к какому хосту идет обращение. Данное поле не входит в число обязательных. Однако оно является **необходимым** в тех случаях, когда одному физическому серверу соответствует несколько виртуальных хостов. В этом поле тогда указывается какой из виртуальных хостов имеется в виду.  
**Host: www.nnov.city.ru**  
  
Примеры запросов:

Простейший запрос:

GET /index.html HTTP/1.0

Посложнее:

GET /somedir/somedoc.html HTTP/1.0

User-Agent: Mozilla/2.0

Accept: text/html

Accept: text/plain

Accept: image/gif

Передача данных CGI- скрипту через метод GET

GET /~paaa/cgi-bin/test.cgi?name=Dmitry&organization=%D3%ED%E8%E2%E5%F0%F1%E8%F2%E5%F2+%CD%E8%E6%ED%E5%E3%EE+%CD%EE%E2%E3%EE%F0%EE%E4%E0&Name=&email=&comment= HTTP/1.0

User-Agent: Mozila/2.0

Accept: text/html

Accept: image/gif

Ответ HTTP-сервера  
Ответ идет от сервера. Состоит он из строки состояния и затем поля ответа Общий заголовок([General-Header](file:///E:\%D0%A3%D1%87%D0%B5%D0%B1%D0%BD%D0%BE%D0%B5%20%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B8%D0%B5%20%D0%BF%D0%BE%20CGI-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8E%20%D0%BE%D1%82%20%D0%9B%D0%B5%D1%88%D0%B8.htm#general_header)) и заголовок тела сообщения ([Entity-Header](file:///E:\%D0%A3%D1%87%D0%B5%D0%B1%D0%BD%D0%BE%D0%B5%20%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B8%D0%B5%20%D0%BF%D0%BE%20CGI-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8E%20%D0%BE%D1%82%20%D0%9B%D0%B5%D1%88%D0%B8.htm#entity_header)),которые уже описаны при обсуждении запроса. и еще идет заголовок ответа(Response-Header).  
Строка состояния имеет следующий формат:  
**HTTP/version <SP> Status-Code <SP> Status-Phrase**  
где HTTP/version версия,Status-Code -3х значный код,и Status-Phrase текстовая фраза, поясняющая код ,пример: HTTP/1.0 200 Ok   
,200 -код означающий успешную обработку запроса,что и поясняет "Ok" Заголовок ответа состоит из полей:  
Location:  
Содержит URI ресурса,может быть использован для переключения клиента в другое место, если например ресурс был перемещен в другое место или на другой сервер.  
Location: http://www.uic.nnov.ru/newlocation/index.html  
  
**Server:**  
Информация о програмном обеспечении сервера  
**Server: Apache/1.1**  
  
**WWW-Autenticate:**  
Параметры аутентификации.  
**WWW-Autenticate: Basic realm="doomsday"**

### Коды ответов HTTP

|  |  |  |
| --- | --- | --- |
| 200 | OK | Ok |
| 201 | Created | Успешная команда POST |
| 202 | Accepted | Запрос принят |
| 203 | Non-Authoritative Information | Запрос GET или HEAD выполнен |
| 204 | No Content | Запрос выполнен но нет содержимого |
| 300 | Multiple Choices | Ресурс обнаружен в нескольких местах |
| 301 | Moved Permanently | Ресурс удален навсегда |
| 302 | Found | Ресурс отстутствует временно |
| 304 | Not Modified | Ресурс не был изменен |
| 400 | Bad Request | Плохой запрос от клиента |
| 401 | Unauthorized | Неавторизованный запрос |
| 402 | Payment Required | Необходима оплата за ресурс |
| 403 | Forbidden | Доступ запрещен |
| 404 | Not Found | Ресурс не найден |
| 405 | Method Not Allowed | Метод не применим для данного ресурса |
| 406 | Not Acceptable | Недопустимый тип ресурса |
| 410 | Gone | Ресурс недоступен |
| 500 | Internal Server Error | Внутренняя ошибка сервера |
| 501 | Not Implemented | Метод не выполнен |
| 502 | Bad Gateway | Неисправный шлюз либо перегруз сервера |

**#!perl**

**#500.cgi**

**print "Status: 500 Internal Server Error\n";**

**print "Content-Type: text/html\n\n";**

**print "<HTML><BODY>";**

**print "<H1>500 Internal Server Error!!</H1>";**

**print "</BODY></HTML>";**