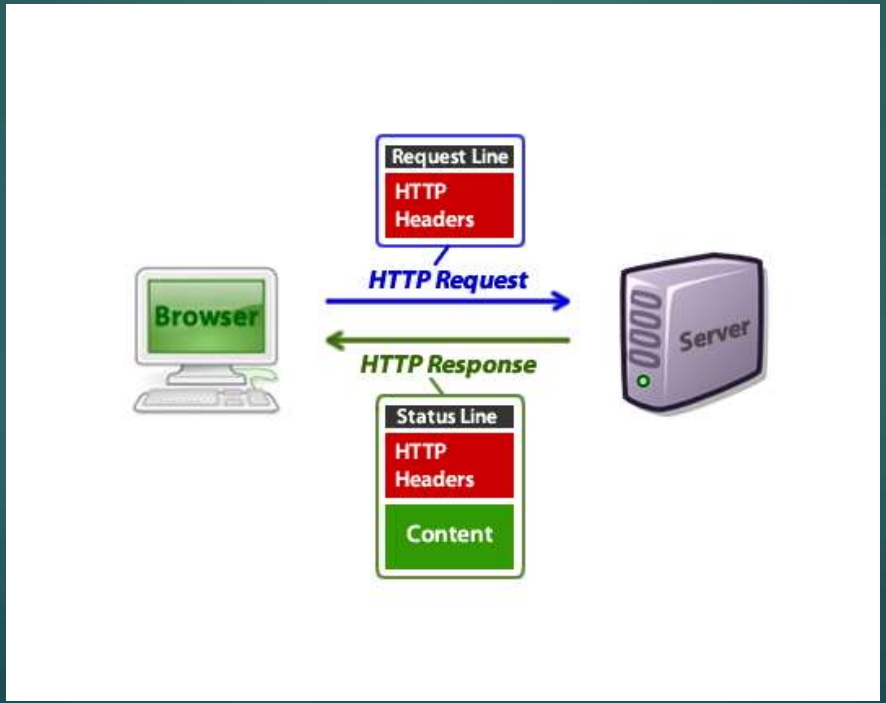
Лекция 3. HTTP-протокол

HTTP

HyperText Transfer Protocol – «гипермәтінді тасымалдау протоколы» – қолданбалы деңгей деректерін тасымалдау протоколы (бастапқыда HTML форматындағы гипермәтіндік құжаттар түрінде, қазіргі уақытта ерікті деректерді тасымалдау үшін пайдаланылады). HTTP негізі «клиент-сервер» технологиясы болып табылады, яғни ол қосылымды бастайтын және сұраныс жіберетін тұтынушылардың (клиенттердің) және сұрауды қабылдау үшін қосылымды күтетін провайдерлердің (серверлердің) болуын болжайды. қажетті әрекеттерді орындап, нәтижесі бар хабарламаны қайтарады

HTTP


Request Line
HTTP Headers
HTTP Request
Browser
Server
HTTP Response
Status Line
HTTP Headers
Content

HTTP

HTTP-дегі манипуляцияның негізгі нысаны клиенттік сұрауда URI (Бірыңғай ресурс идентификаторы) арқылы көрсетілген ресурс болып табылады. Әдетте, бұл ресурстар серверде сақталған файлдар, бірақ олар логикалық нысандар немесе дерексіз нәрсе болуы мүмкін. HTTP протоколының ерекшелігі сұрауда және жауапта бір ресурстың әртүрлі параметрлермен қалай ұсынылатынын көрсету мүмкіндігі болып табылады: пішім, кодтау, тіл және т.б. (атап айтқанда, бұл үшін HTTP тақырыбы қолданылады.) және сервер екілік деректерді алмастыра алады, бірақ бұл протокол мәтіндік

URI.

URI - сұраққа жауап береді: «Бір нәрсені қайдан және қалай табуға болады?»

Мысалы: `http://example.com/just/some/long/path/path`

HTTP. Тарихы

HTTP/0.9

HTTP 1991 жылы наурызда Тим Бернерс-Ли, содан кейін CERN-де Интернеттегі құжаттарға қол жеткізу және гипермәтінді пайдалану арқылы шарлауды жеңілдету механизмі ретінде ұсынылды. HTTP/0.9 протоколының ең алғашқы нұсқасы алғаш рет 1992 жылдың қаңтарында жарияланды (бірақ енгізу 1990 жылдан басталады). Протокол спецификациясы HTTP клиенттері мен серверлері арасындағы өзара әрекеттесу ережелерін оңтайландыруға, сондай-ақ екі құрамдас арасындағы функцияларды нақты бөлуге әкелді. Негізгі синтаксистік және семантикалық ережелер құжатталды.

HTTP/1.0

1996 жылдың мамырында HTTP/1.0 құрамдастарының көпшілігін іске асыру үшін негіз болған HTTP-ті практикалық енгізу үшін RFC 1945 шығарылды.

HTTP/1.1

1999 жылы маусымда қабылданған хаттаманың қазіргі нұсқасы. Бұл шығарылымдағы жаңалық «тұрақты қосылым» режимі болды: TCP қосылымы сұрауға жауап жіберілгеннен кейін ашық күйінде қалуы мүмкін, бұл бір қосылымда бірнеше сұрауларды жіберуге мүмкіндік береді. Клиент енді ортақ хостингті оңай ұйымдастыруға мүмкіндік беретін, ол қатынасатын хосттың аты туралы ақпаратты жіберуі қажет.

HTTP/2

2015 жылғы 11 ақпанда хаттаманың келесі нұсқасы жобасының соңғы нұсқалары жарияланды. Алдыңғы нұсқалардан айырмашылығы, HTTP/2 протоколы екілік болып табылады. Негізгі мүмкіндіктерге сұрауды мультиплекстеу, сұрау басымдылығын анықтау, тақырыпты сығу, бір TCP қосылымы арқылы параллель бірнеше элементтерді жүктеу және сервер тарапынан проактивті push хабарландыруларына қолдау кіреді.

RFC.

Дүниежүзілік желіде кеңінен қолданылатын техникалық сипаттамалар мен стандарттарды қамтитын құжаттар.

5000-нан астам құжат бар.

HTTP. Структура

Әрбір HTTP хабарламасы тізімде көрсетілген ретпен жіберілетін үш бөліктен тұрады:

Бастапқы жол – хабарлама түрін анықтайды;
Тақырыптар – хабарлама мәтінін, жіберу параметрлерін және басқа ақпаратты сипаттайды;
Хабардың негізгі бөлігі хабарлама деректерінің өзі болып табылады. Тақырыптардан бос жолмен бөлінуі керек.

Хабардың үстіңгі деректемелері мен мәтіні өткізілмеуі мүмкін, бірақ бастапқы жол сұрау/жауап түрін көрсететіндіктен, қажетті элемент болып табылады. Ерекшелік протоколдың 0.9 нұсқасы болып табылады, онда сұрау хабарламасы тек бастапқы жолды, ал жауап хабарламасында тек хабарлама мәтінін қамтиды.

Протоколдың 1.1 нұсқасы үшін сұрау хабарында хост тақырыбы болуы керек.

HTTP. Структура. Стартовая строка запроса

Метод URI HTTP/Версия

HTTP. Структура. Стартовая строка запроса

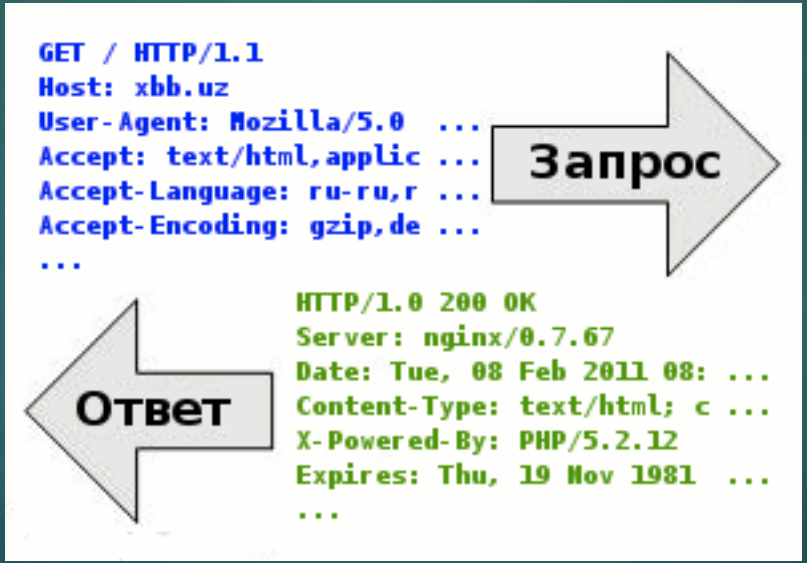
Әдіс (ағыл. Method) - сұрау атауы, бас әріппен бір сөз
URI сұралған құжатқа жолды көрсетеді.
Нұсқа (ағыл. Version) – нүкте арқылы бөлінген сандар жұбы. Мысалы: 1.0

HTTP. Структура. Стартовая строка ответа

HTTP/нұсқа күй-код түсіндірмесі

Нұсқа – сұраудағыдай нүктемен бөлінген сандар жұбы.
Күй коды - үш сан. Күй коды хабарламаның әрі қарай мазмұнын және клиенттің әрекетін анықтайды.
Түсіндіру (ағыл. Reason Phrase) – пайдаланушыға жауап кодының мәтіндік қысқаша түсіндірмесі. Хабарламаға ешқандай әсер етпейді және міндетті емес.

HTTP. Структура



HTTP. Методы

Ресурстағы негізгі әрекетті көрсететін басқару таңбалары мен бөлгіштерден басқа кез келген таңбалар тізбегі. Method - бас әріппен жазылатын қысқа ағылшын сөзі. Әдіс атауы регистрді ескереді

HTTP. Методы

Әрбір сервер кем дегенде GET және HEAD әдістерін қолдауы керек. Егер сервер клиент көрсеткен әдісті танымаса, ол 501 (Орындалмаған) күйін қайтаруы керек. Егер әдіс серверге белгілі болса, бірақ ол белгілі бір ресурсқа қолданылмаса, 405 (әдіске рұқсат етілмеген) хабарламасы қайтарылады. Екі жағдайда да сервер жауап хабарында қолдау көрсетілетін әдістер тізімі бар Рұқсат ету тақырыбын қамтуы КЕРЕК.

GET және HEAD әдістерінен басқа POST әдісі жиі қолданылады.

HTTP. Методы

OPTIONS
GET
HEAD
POST
PUT
PATCH
DELETE
TRACE
CONNECT

HTTP. Методы

OPTIONS
Белгілі бір ресурс үшін веб-сервер мүмкіндіктерін немесе қосылым опцияларын анықтау үшін пайдаланылады. Жауап ретінде сервер қолдау көрсетілетін әдістер тізімі бар Рұқсат беру тақырыбын қамтуы КЕРЕК. Жауап тақырыбы да қолдау көрсетілетін кеңейтімдер туралы ақпаратты қамтуы мүмкін.
Бұл әдісті орындау нәтижесі кэштелмейді

HTTP. Методы

GET
Көрсетілген ресурстың мазмұнын сұрау үшін пайдаланылады. Сондай-ақ, процесті GET әдісі арқылы бастауға болады. Бұл жағдайда процестің барысы туралы ақпарат жауап хабарламасының негізгі бөлігінде болуы керек.

Клиент сұрауды орындау параметрлерін мақсатты ресурстың URI мекенжайында «?» таңбасынан кейін жібере алады:
/path/resource?param1=value1&param2=value2 HTTP/1.1 АЛУ

HTTP стандартына сәйкес GET сұраулары идемпотентті болып саналады.

HTTP. Методы

HEAD
GET әдісіне ұқсас, тек сервер жауапында дененің болмауы. HEAD сұрауы әдетте метадеректерді шығарып алу, ресурстың бар-жоғын тексеру (URL тексеруі) және соңғы қол жеткізгеннен бері оның өзгергенін көру үшін пайдаланылады.

Жауап тақырыптарын кэштеуге болады. Ресурс метадеректері кэштегі сәйкес ақпаратқа сәйкес келмесе, ресурс көшірмесі ескірген деп белгіленеді

HTTP. Методы

POST
Пайдаланушы деректерін берілген ресурсқа жіберу үшін пайдаланылады. Мысалы, блогтарда келушілер әдетте жазбалардағы пікірлерін HTML пішініне енгізе алады, содан кейін олар POST әдісі бойынша серверге жіберіледі және ол оларды бетке орналастырады. Бұл жағдайда жіберілген деректер (блог мысалында, түсініктеме мәтіні) сұрау мәтініне қосылады. Сол сияқты POST әдісі әдетте файлдарды серверге жүктеп салу үшін қолданылады.

GET әдісінен айырмашылығы, POST әдісі идемпотентті болып саналмайды, яғни бір POST сұрауларының қайталануы әртүрлі нәтижелерді беруі мүмкін (мысалы, әрбір түсініктеме жіберілгеннен кейін осы түсініктеменің басқа көшірмесі пайда болады).

Нәтиже 200 (Жарайды) болса, жауап органы сұрау нәтижесі туралы хабарламаны қамтуы керек. Ресурс жасалған болса, сервер Орын тақырыбындағы жаңа ресурстың URI коды бар 201 (Жасалған) жауабын қайтаруы КЕРЕК.

POST әдісін орындау үшін сервердің жауап хабары кэштелмеген

HTTP. Методы

PUT
Сұрау мазмұнын сұрауда көрсетілген URI мекенжайына жүктеп алу үшін пайдаланылады. Берілген URI үшін ресурс жоқ болса, сервер оны жасайды және 201 (Жасалған) күйін қайтарады. Ресурс өзгертілсе, сервер 200 (Жарайды) немесе 204 (Мазмұн жоқ) мәнін қайтарады. Сервер хабарламамен клиент жіберген жарамсыз Content-* тақырыптарын елемеуге КЕРЕК. Осы тақырыптардың кез келгенін тану мүмкін болмаса немесе ағымдағы шарттарда жарамсыз болса, 501 (Орындалмаған) қате коды қайтарылуы керек.

POST және PUT әдістерінің арасындағы түбегейлі айырмашылық URI ресурсының мақсатын түсінуде жатыр. POST әдісі клиент жіберген мазмұн көрсетілген URI мекенжайында өңделетінін болжайды. PUT пайдалану арқылы клиент жүктелетін мазмұн берілген URI мекенжайындағы ресурсқа сәйкес келеді деп болжайды.

PUT әдісіне сервердің жауап хабарлары кэштелмейді

HTTP. Методы

PATCH
PUT сияқты, бірақ тек ресурс фрагментіне қолданылады

HTTP. Методы

DELETE
Көрсетілген ресурсты жояды

HTTP. Методы

TRACE
Клиент сұрауда аралық серверлер қандай ақпаратты қосатынын немесе өзгертетінін көре алатындай қабылданған сұрауды қайтарады

HTTP. Методы

CONNECT
Әдетте шифрланбаған прокси арқылы қауіпсіз SSL қосылымын орнатуды жеңілдету үшін сұрау қосылымын мөлдір TCP/IP туннелыіне түрлендіреді.

HTTP. Коды состояния

1xx Ақпараттық
Бұл сыныпта жіберу процесі туралы ақпарат беретін кодтар бар. HTTP/1.0 жүйесінде осы кодтары бар хабарларды елемеу керек. HTTP/1.1-де клиент бұл хабар класын қалыпты жауап ретінде қабылдауға дайын болуы керек, бірақ серверге ештеңе жіберудің қажеті жоқ. Серверден келген хабарлардың өзі тек жауаптың бастапқы жолын және қажет болса, жауапқа арналған бірнеше тақырып өрістерін қамтиды. Прокси-серверлер мұндай хабарламаларды серверден клиентке әрі қарай жіберуі керек.

2xx Сәттілік
Бұл сыныптың хабарламалары клиент сұранысын сәтті қабылдау және өңдеу жағдайлары туралы хабарлайды. Күйге байланысты сервер тақырыптарды және хабарлама мәтінін жіберуі мүмкін.

3xx қайта бағыттау
3xx сынып кодтары клиентке операцияны сәтті аяқтау үшін басқа сұрау (әдетте басқа URI арқылы) жасалуы керектігін айтады. Осы сыныптың бес кодтары 301, 302, 303, 305 және 307 тікелей бағыттауға (қайта бағыттауға) қатысты. Клиент сұрау салуы тиіс мекенжайды Орын тақырыбындағы сервер көрсетеді. Бұл мақсатты URI-де фрагменттерді пайдалануға мүмкіндік береді.

4xx клиент қатесі
4xx код класы клиент жағындағы қателерді көрсетуге арналған. HEAD-тан басқа барлық әдістерді пайдаланған кезде сервер хабарламаның негізгі бөлігіндегі пайдаланушы үшін гипермәтіндік түсініктемені қайтаруы керек.

5xx сервер қатесі
5xx кодтары сервердің кінәсінен сәтсіз жұмыс жағдайлары үшін бөлінген. HEAD әдісін қолданудан басқа барлық жағдайлар үшін сервер клиент пайдаланушыға көрсететін хабарламаның негізгі бөлігінде түсіндірмені қамтуы МІНДЕТТІ.

HTTP. Коды состояния

1xx: Informational (информационные)
100 Continue («продолжай»)
101 Switching Protocols («переключение протоколов»)
102 Processing («идёт обработка»)

HTTP. Коды состояния

2xx: Success (успешно)

- 200 OK («хорошо»)
- 201 Created («создано»)
- 202 Accepted («принято»)
- 203 Non-Authoritative Information («информация не авторитетна»)
- 204 No Content («нет содержимого»)
- 205 Reset Content («сбросить содержимое»)
- 206 Partial Content («частичное содержимое»)
- 207 Multi-Status («многостатусный»)

HTTP. Коды состояния

3xx: Redirection (перенаправление)
300 Multiple Choices («множество выборов»)
301 Moved Permanently («перемещено навсегда»)
302 Moved Temporarily («перемещено временно»)
302 Found («найдено»)
303 See Other (смотреть другое)
304 Not Modified (не изменялось)
305 Use Proxy («использовать прокси»)
306 — зарезервировано (код использовался только в ранних спецификациях)
307 Temporary Redirect («временное перенаправление»)

HTTP. Коды состояния

4xx: Client Error (ошибка клиента)
400 Bad Request («плохой, неверный запрос»)
401 Unauthorized («не авторизован»)
402 Payment Required («необходима оплата»)
403 Forbidden («запрещено»)
404 Not Found («не найдено»)
405 Method Not Allowed («метод не поддерживается»)
406 Not Acceptable («неприемлемо»)
407 Proxy Authentication Required («необходима аутентификация прокси»)
408 Request Timeout («истекло время ожидания»)
409 Conflict («конфликт»)
410 Gone («удалён»)
411 Length Required («необходима длина»)
412 Precondition Failed («условие ложно»)

HTTP. Коды состояния

4xx: Client Error (ошибка клиента)
413 Request Entity Too Large («размер запроса слишком велик»)
414 Request-URI Too Large («запрашиваемый URI слишком длинный»)
415 Unsupported Media Type («неподдерживаемый тип данных»)
416 Requested Range Not Satisfiable («запрашиваемый диапазон не достижим»)
417 Expectation Failed («ожидаемое неприемлемо»)
422 Unprocessable Entity («необрабатываемый экземпляр»)
423 Locked («заблокировано»)
424 Failed Dependency («невыполненная зависимость»)
425 Unordered Collection («неупорядоченный набор»)
426 Upgrade Required («необходимо обновление»)
428 Precondition Required («необходимо предусловие»)
429 Too Many Requests («слишком много запросов»)
431 Request Header Fields Too Large («поля заголовка запроса слишком большие»)
434 Requested host unavailable. («Запрашиваемый адрес недоступен»)
444 Закрывает соединение без передачи заголовка ответа. Нестандартный код
449 Retry With («повторить с»)
451 Unavailable For Legal Reasons («недоступно по юридическим причинам»)

HTTP. Коды состояния

5xx: Server Error (ошибка сервера)
500 Internal Server Error («внутренняя ошибка сервера»)
501 Not Implemented («не реализовано»)
502 Bad Gateway («плохой, ошибочный шлюз»)
503 Service Unavailable («сервис недоступен»)
504 Gateway Timeout («шлюз не отвечает»)
505 HTTP Version Not Supported («версия HTTP не поддерживается»)
506 Variant Also Negotiates («вариант тоже проводит согласование»)
507 Insufficient Storage («переполнение хранилища»)
508 Loop Detected («обнаружена петля»)
509 Bandwidth Limit Exceeded («исчерпана пропускная ширина канала»)
510 Not Extended («не расширено»)
511 Network Authentication Required («требуется сетевая аутентификация»)

HTTP. Заголовки

HTTP тақырыптары қос нүктемен бөлінген параметр-мән жұбын қамтитын HTTP хабарындағы жолдар. Тақырып пішімі ARPA жалпы мәтіндік хабар тақырыбы пішіміне (RFC 822) сәйкес келеді. Тақырыптар хабарлама мәтінінен кем дегенде бір бос жолмен бөлінуі керек

HTTP. Заголовки

```
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
```

HTTP. Заголовки

Барлық тақырыптар төрт негізгі топқа бөлінеді:

Жалпы тақырыптар - кез келген клиенттік және серверлік хабарламаға қосылуы мүмкін.
Сұраныс тақырыптары - тек клиент сұрауларында қолданылады.
Жауап тақырыптары - тек серверден жауаптар үшін.
Entity Headers («Нысанның тақырыптары») – хабарламаның әрбір нысанымен бірге жүреді

HTTP. Заголовки

Заголовок	Описание	Пример
Accept	Список допустимых форматов ресурса	Accept: text/plain
Accept-Charset	Перечень поддерживаемых кодировок для предоставления пользователю	Accept-Charset: utf-8
Allow	Список поддерживаемых методов	Allow: OPTIONS, GET, HEAD
Referer	URI ресурса, после которого клиент сделал текущий запрос	Referer: http://en.wikipedia.org/wiki/Main_Page
User-Agent	Список названий и версий клиента и его компонентов с комментариями	User-Agent: Mozilla/5.0 (X11; Linux i686; rv:2.0.1) Gecko/20100101 Firefox/4.0.1

HTTP vs HTTPs

HyperText Transfer Protocol Secure — шифрлауды қолдайтын HTTP протоколының кеңейтімі. HTTPS протоколы арқылы тасымалданатын деректер SSL немесе TLS криптографиялық протоколында «оралған».
HTTP-ден айырмашылығы, HTTPS стандартты TCP порты 443 болып табылады

HTTP vs HTTPs

HTTPS жеке протокол емес. Бұл SSL және TLS шифрланған тасымалдау механизмдері арқылы жұмыс істейтін қарапайым HTTP. Ол шифрлау құралдары пайдаланылған және сервер сертификаты тексерілген және сенімді болған жағдайда, желіні иіскеуге негізделген шабуылдардан қорғауды қамтамасыз етеді.

HTTP vs HTTPs

Әдепкі бойынша, HTTPS URL мекенжайы 443 TCP портын пайдаланады (қорғалмаған HTTP үшін 80). Веб-серверді https қосылымдарын өңдеуге дайындау үшін әкімші жүйеде сол веб-серверге сертификат алып, орнатуы керек. Сертификат 2 бөліктен (2 кілт) тұрады - ашық және жеке. Сертификаттың жалпы бөлігі қауіпсіз қосылымда клиенттен серверге трафикті шифрлау үшін пайдаланылады, жеке бөлігі серверде клиенттен алынған шифрланған трафикті шешу үшін қолданылады. Жеке/ашық кілттер жұбы жасалғаннан кейін ашық кілт негізінде сертификаттау орталығына сертификат сұрауы жасалады, оған жауап ретінде CA қол қойылған сертификатты жібереді. Қол қою кезінде CA клиентті тексереді, бұл сертификат иесінің өзі мәлімдеген адам екеніне кепілдік беруге мүмкіндік береді (әдетте бұл ақылы қызмет).

HTTP vs HTTPs

Дәстүрлі түрде бір IP мекенжайында тек бір HTTPS сайты жұмыс істей алады. Әртүрлі сертификаттары бар бірнеше HTTPS сайттары Server Name Indication (SNI) деп аталатын TLS кеңейтімін пайдаланады.

HTTP Cookies

- **HTTP Cookie (cookie файлдары) – веб-сервер жіберген және клиент браузерінде сақталған мәтіндік деректердің шағын бөлігі. Сәйкес сайттың беті ашылған сайын браузер сақталған деректер фрагментін HTTP тақырыптары арқылы веб-серверге қайта жібереді..**

HTTP Cookies

Cookie файлдары мыналар үшін пайдаланылады:

пайдаланушының аутентификациясы;

жеке қалаулар мен пайдаланушы параметрлерін сақтау;

пайдаланушының кіру сеансының күйін қадағалау;

пайдаланушылар туралы статистиканы жүргізу.

Установка Cookie

- В веб-сервердің HTTP жауап тақырыбы браузерге cookie файлдарын сақтау туралы нұсқауды қамтуы мүмкін :

```
HTTP/1.1 200 OK
Content-Type: text/html
Set-Cookie: name=value

Содержимое страницы
```

HTTP Cookies

- **HTTP Cookie (cookie файлдары) – веб-сервер жіберген және клиент браузерінде сақталған мәтіндік деректердің шағын бөлігі. Сәйкес сайттың беті ашылған сайын браузер сақталған деректер фрагментін HTTP тақырыптары арқылы веб-серверге қайта жібереді.**

Установка Cookie

- Set-Cookie жолы әдетте HTTP жауабына HTTP серверінің өзі емес, онымен бірге жұмыс істейтін CGI бағдарламасы арқылы қосылады. HTTP сервері браузерге мұндай бағдарламаның нәтижесін ғана жібереді.

Чтение Cookie

- Set-Cookie жолы сервер браузерге cookie файлын сақтауды қалағанкезде ғана жіберіледі. Бұл жағдайда браузер name=value жолын есте сақтайды және оны әрбір келесі сұраумен серверге жібереді..

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: name=value
```

- Cookie мәнін сервер "Set-Cookie" файлын қайта жіберу арқылы өзгертуі мүмкін.

Атрибуты Cookie

- Кроме пары «имя/значение» куки может содержать **срок действия**, **путь** и **доменное имя**, на которое оно распространяется. Пример:

```
Set-Cookie: name=newvalue; expires=date;
path=/; domain=.example.org.
```

Атрибуты Cookie

- Домен и путь говорят браузеру, что куки нужно отправлять обратно на сервер при запросах URL для указанного домена и пути. Если они не указаны, используются домен и путь запрошенной страницы.
- Дата истечения указывается в формате «Нед, ДД Мес ГГГГ ЧЧ:ММ:СС GMT». Например:
- `Set-Cookie: RMID=732423sdfs73242; expires=Fri, 31 Dec 2013 23:59:59 GMT; path=/; domain=.example.net`

Типы Cookie

- **Куки сессии** – существует только на то время, пока пользователь производит навигацию по сайту. Куки сессии создаётся автоматически, если не указан срок действия куки.
- **Постоянные куки** – существует до тех пор, пока не закончится срок действия куки. Например, если куки имеет атрибут Max-Age установленный на 1 год (например), то значение Cookie будет отправляться браузером на Web-сервер при каждом обращении в течение года.

Безопасность Cookie

- Куки легко перехватить и подменить (например, для получения доступа к учетной записи), если пользователь использует нешифрованное соединение с сервером.

Способы задания Cookie

1) Через клиентский JavaScript
2) Через прямую установку HTTP-заголовков на сервере

Cookie в CGI

- Получение Cookie в среде CGI происходит с помощью переменной окружения **HTTP_COOKIE**, которая в точности повторяет HTTP-заголовок клиента «**Cookie**».
- Формат Cookie имеет следующий вид:

```
name=value; name2=value2
```

Cookie в PHP

- Любые cookies, отправленные серверу браузером клиента, будут автоматически включены в суперглобальный массив **`$_COOKIE`**

setcookie()

- Задает cookie, которое будет передано клиенту вместе с другими HTTP заголовками.
- Как и любой другой заголовок, cookie должны передаваться **до** того как будут выведены какие-либо другие данные скрипта (ограничение протокола).
- Все аргументы, за исключением **`name`**, являются необязательными. Если нужно пропустить какой-либо аргумент, можно вместо него поставить пустую строку ("") или ноль (0) для **`expire`**.

setcookie()

- **`name`** – Наименование cookie.
- **`value`** – Значение cookie.
- **`expire`** – Время, когда срок действия cookie истекает.
  - Если задать 0 или пропустить этот аргумент, срок действия cookie истечет с окончанием сессии (при закрытии браузера).
- **`path`** – Путь к директории на сервере, из которой будут доступны cookie.
  - Если задать '/', cookie будут доступны во всем домене domain.
  - Если задать '/foo/', cookie будут доступны только из директории /foo/ и всех ее поддиректорий (например, /foo/bar/) домена domain.
  - По умолчанию значением является текущая директория, в которой cookie устанавливается.

setcookie()

- **`domain`** – Домен, которому доступны cookie.
  - Задание домена 'www.example.com' сделает cookie доступными в поддомене www и поддоменах более высоких порядков. Cookie доступные низким уровням, таким как 'example.com', будут доступны во всех поддоменах высших уровней, с том числе 'www.example.com'.
- **`secure`** – Указывает на то, что значение cookie должно передаваться от клиента по защищенному HTTPS соединению.
  - Если задано TRUE, cookie от клиента будет передано на сервер, только если установлено защищенное соединение. При передаче cookie от сервера клиенту следить за тем, чтобы cookie этого типа передавались по защищенному каналу, должен программист веб-сервера.
- **`httponly`** – Если задано TRUE, cookie будут доступны только через HTTP протокол.
  - В этом случае cookie не будут доступны скриптовым языкам, вроде JavaScript.

setcookie()

- Если перед вызовом функции клиенту уже передавался какой-либо вывод (тэги, пустые строки, пробелы, текст и т.п.), setcookie() вызовет отказ и вернет **FALSE**.
- Если setcookie() успешно отработает, то вернет **TRUE**.
  - НО! Это не означает, что клиентское приложение (браузер) правильно приняло и обработало cookie.

Пример

```php
<?php
$value = 'что-то где-то';

setcookie("TestCookie", $value);

setcookie("TestCookie", $value, time()+3600)
;  /* срок действия 1 час */

setcookie("TestCookie", $value, time()+3600,
 "/~rasmus/", "example.com", 1);
?>
```

Вывод cookie

```php
<?php
// Вывести одно конкретное значение cookie
echo $_COOKIE["TestCookie"];

// Устанавливаем Cookie до конца сессии:
// В случае успешной установки Cookie, функция SetCookie возвращает TRUE:
if (SetCookie("Test","Value")) echo "<h3>Cookies успешно установлены!</h3>";
/*
  В целях тестирования и отладки может
  пригодиться вывод всех cookie
*/
print_r($_COOKIE);
?>
```

Удаление cookies

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

```
<?php
// установка даты истечения срока действия
на час назад
setcookie ("TestCookie", "", time() -
 3600);
setcookie ("TestCookie", "", time() -
 3600, "/~rasmus/", "example.com", 1);
?>
```

setcookie() и массивы

Для того, чтобы использовать каждому cookie нужно дать имя в соответствии с правилами именования массивов. Такая возможность позволяет поместить столько значений, сколько имеется элементов в массиве.

```php
<?php
// отправка cookie
setcookie("cookie[three]", "cookiethree");
setcookie("cookie[two]", "cookietwo");
setcookie("cookie[one]", "cookieone");

// после перезагрузки страницы, выведем cookie
if (isset($_COOKIE['cookie'])) {
    foreach ($_COOKIE['cookie'] as $name => $value) {
        $name = htmlspecialchars($name);
        $value = htmlspecialchars($value);
        echo "$name : $value <br />\n";
    }
}
?>
```

Лабораторная работа

- Сделать браузерную мини-игру
- При первом заходе на страницу спросить логин
  - Записать логин в Cookie
  - Приветствовать уже представленных посетителей
  - Сделать возможным удаление информации о посетителе (ручной сброс)
- Реализовать при помощи форм интерактивность
  - Обеспечить сохранение информации
  - Обеспечить безопасность (не допускать взлом игровых параметров)

Клиент-сервер архитектура жүйесі

"Клиент-сервер" жүйелік архитектурасының негізгі принциптері мыналар: АЖ- лер желінің әр түрлі шетіндегі

болып 2-ге бөлінеді

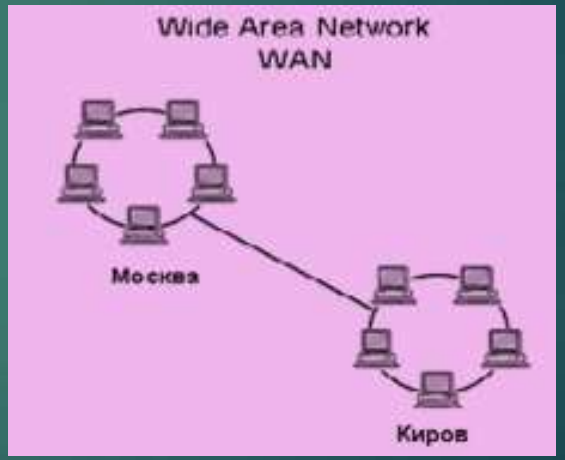
Серверлік жақ өзімен қоса қуатты аппараттандырылған құралы, қажетті стандартты бағдарламалық қамтамасыз етуі, деректер қорын басқару жүйесі және ДҚ бар арнайы мамандандырылған аппараттандырылған комплексте

Қолданбалы программалар

Қолданбалы программа немесе пайдаланушы жүйенің клиенттік бөлімімен әрекеттеседі. Қосымшаның клиенттік жағы пайдаланушының жұмыс орнында жұмыс істейді.

Жүйенің клиенттік бөлімі қажеттілік жағдайында серверлік бөлімге желі (локальді немесе глобальді) бойынша байланыс жасайды. Сонымен бірге клиент пен серверге қарағанда қатынас айқын орындалады.

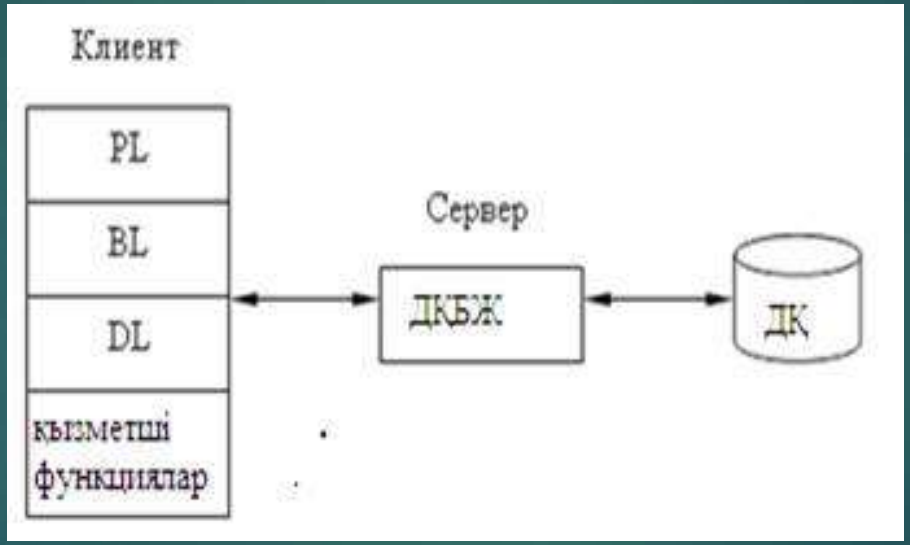
Локальді желі



Глобальді желі

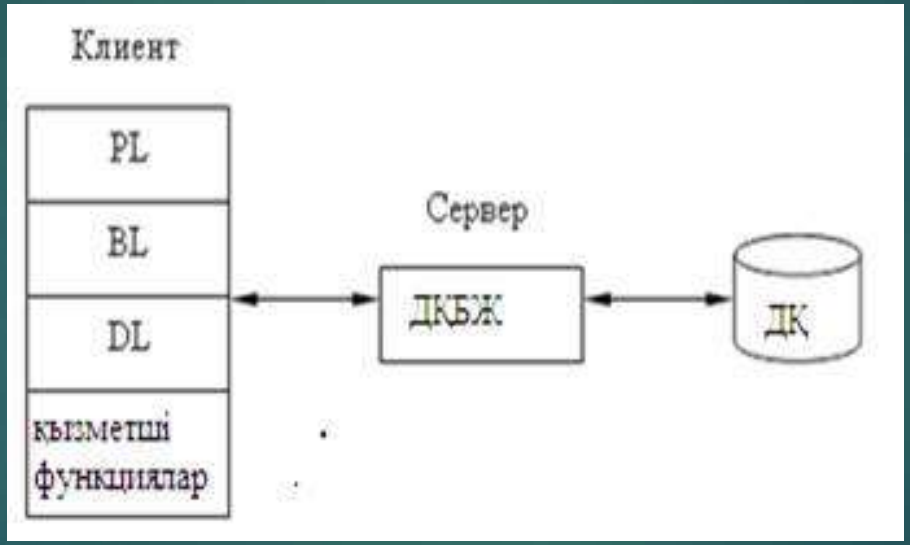
Бұл қатынасты орындайтын желілік компоненттің өзімен бірге жиынтығында қажетті желілік жабдықтар, желілік түйіндер арасындағы байланыстарды қамтамасыз ететін бағдарламалық технологиялық жиынтық және де сұраныстар мен олардың орындалу нәтижелерінің айырбасы үшін интерфейстік бағдарламалық қабаттар (протокол немесе протоколдар) бар. Клиенттік және серверлік бөлімдердің арасында SQL деректер қорының тілі негізгі интерфейс болып табылады.

процесстердің: клиент және сервер арасындағы есептеуіш жүктеменің бөлінуі.



«Клиент-сервер» қосымшасының типтік құрылымы

процесстердің: клиент және сервер арасындағы есептеуіш жүктеменің бөлінуі.



«Клиент-сервер» қосымшасының типтік құрылымы

«Интеллектуальды» клиенттер- «клиент-серверлік» қосымшасы ұйымдастырудың ең көп таралған әдісі. «Интеллектуалдық» клиентке ұсыныстың сервисінің орындайтынына, бизнес- логиканы және деректерді басқаруды сенуге болады.

Вопросы и ответы