Введение в HTTP протокол

Здесь мы попробуем прикоснуться к огромной области программирования, связанной с поддержкой различных интернет сервисов. Естественно, нам, в основном, придется ограничиться упоминанием некоторых технологий и рассмотреть самые примитивные подходы.

Общая задача — обеспечить передачу информационных материалов с одной сетевой точки (информационной базы данных) на другие сетевые точки (клиентские станции) по запросам от этих клиентов, и также отобразить эти результаты в удобной для человека форме.

В силу общности и важности этой задачи для ее решения было разработано множество технологий, которые продолжают развиваться и в настоящее время. Упомянем некоторые из них.

TCP — базовый транспортный протокол. Этот протокол естественным образом был положен в основу необходимых реализаций, поскольку он обеспечивает надежность передачи данных.

HTML — HyperText Markup Language. Язык гипертекстовой разметки. Был создан для того, чтобы размечать компоненты передаваемой информации (web-странички) с целью их подходящего (правильного, красивого, удобного и т.п.) отображения для пользователя на клиентской машине. За время своего существования разросся до огромных объемов и породил множество дополнительных концепций и технологий, связанных с обработкой и представлением данных, передаваемых на клиентские машины.

HTTP — HyperText Transfer Protocol. Это протокол, предназначенный изначально специально для пересылки и обработки HTML документов. Интернет браузеры (их много разных) собственно и занимаются передачей и приемом HTTP пакетов, которые поддерживают стандартную идеологию работы клиентов и сервера, т.е. клиент отправляет серверу HTTP запрос на некоторые данные и получает от сервера HTTP ответ, который в общем случае содержит затребованный HTML документ. Затем браузер отображает этот документ в своих окнах в соответствии с указанной разметкой.

Этих трех компонент уже хватает, чтобы построить простейший Web сервис с нуля и “на коленке”, хотя в реальной практике, конечно, используются разнообразные инструментальные среды и типовые решения.

Помимо этого с развитием информационных и интернет технологий появились и другие инструменты для поддержки решения данной задачи. Можно упомянуть, например, Javascript, CSS, JSON и другие, о которых мы скажем здесь, разве что, пару слов.

Рассмотрим теперь эти понятия чуть более подробно.

URL — Uniform Resource Locator

Рассматриваемая нами общая задача преподагает получение информации от некоторого интернет ресурса. Таким образом, возникает проблема идентификации этого ресурса. Для этой цели было введено понятие URL как некоторой унифицированной записи, которая показывает как и откуда можно получить требуемые данные (т.е. местоположение и способ получения). В общем виде конструкцию URL можно записать в виде

`<протокол>://<логин>:<пароль>@<хост>:<порт>/<URL-путь>?<параметры>\#<якорь>`

Здесь

`<протокол>` — протокол, который дает возможность получить ресурс;

`<логин>:<пароль>` — идентицикация пользователя и его прав, если она предусмотрена для доступа к ресурсу;

`<хост>:<порт>` — сетевая идентификация станции, отвечающей за данный ресурс;

`<URL-путь>` — относительный путь к файлу, обеспечивающему ресурс на данной станции (например, к html файлу, или к программе, запуск которой предоставит нам ресурс);

<параметры> — параметры, которые могут быть необходимы для получения ресурса, они задаются в виде последовательности записей <имя>=<значение>;

<якорь> — метка, указывающая на отдельную часть ресурса, если он поделен на части.

Обычно не все из этих полей необходимы, к том смысле, что если они не указаны, то при работе используется некоторое значение по умолчанию. Например, для протокола значение по умолчанию — http, для пользователя — anonimous с пустым паролем, т.е. общедоступные данные, для порта — тот порт, который назначен данному сервису по умолчанию (например, 80 для http), для URL-пути — файл index.html и т.д. Тут надо понимать, что подставлять значения по умолчанию должна та программа, которая такой URL обрабатывает, т.е. если мы реализуем свой собственный HTTP сервер, то это будет нашей задачей.

HTTP — HyperText Transfer Protocol

HTTP протокол формирует пакеты двух типов — запрос (от клиента к серверу) и ответ (от сервера к клиенту). В обоих случаях данные, которые требуется переслать, снабжаются специальным заголовком, и эта пара (заголовок+данные) пересылаются средствами TCP протокола.

Заголовок HTTP пакета представляет собой набор текстовых строк, которые отделеные от последующих данных пустой строкой. Таким образом, заголовок можно легко прочитать, если его распечатать, так как он содержит обычный текст.

Первая строка в заголовке запроса указывает на то, что и как хочется получить клиенту, последующие строки задают рекомендации серверу или возможную дополнительную информацию, которую сервер мог бы использовать для формирования ответа.

Первая строка в заголовке ответа сервера содержит информацию об успешности обработки запроса, последующие строки заголовка также содержат дополнительную информацию или рекомендации, которые сервер хочет сообщить клиенту.

Данные, идущие после заголовка в запросе или ответе, могут присутсвовать или отсутствовать в зависимости от смысла выполняемых действий.

По историческим причинам текстовые строки заголовка завершаются в стиле Windows, т.е. содержат два байта — возврат каретки и перевод строки, которые мы будем обозначать <CR> <LF> (carridge return, line feed, или в обозначениях языков C/C++ как $\backslash r\backslash n$).

HTTP, формат запроса

```
метод  URL  версия <CR> <LF>                          --- request line
название заголовка : значение <CR> <LF>               --- header line
.......                                               --- header line
<CR> <LF>
entity body                                           --- данные, если надо
```

Метод — это “команда”, которую должен выполнить сервер. В нашем случае мы рассматриваем два возможных метода — GET и POST. Остальные строки задают параметры, которые сервер может принять во внимание при формировании наиболее подходящего ответа (а может и не принять и поступить по-своему). В методе GET главные параметры запроса должны быть указаны в поле URL. В этом случае длина поля URL не должна превышать 1 килобайта. Если же данные для запроса, отправляемого на сервер, занимают много места, то можно использовать метод POST, в котором эти данные передаются через поле entity body. Например, если мы ходим получить от сервера файл, расположенный по относительному пути /dir/file.html, то можно сформировать такой запрос

```
GET  /dir/file.html HTTP/1.1 <CR> <LF>                --- GET по протоколу HTTP/1.1
Host:  localhost:5555 <CR> <LF>                       --- к кому обращаемся
```

```
Connection : keep-alive <CR> <LF>                          --- или close
User-agent : Mozilla Firefox 27.0.1 <CR> <LF>              --- это наш браузер
Accept : text/html  image/gif image/jpeg <CR> <LF>         --- что ожидаем в ответе
Accept-language : ru <CR> <LF>                             --- желательный тип кодировки
<CR> <LF>
```

На самом деле подобный запрос формирует браузер, и задача сервера — правильно на него ответить.

HTTP, формат отклика

```
версия код-отклика сообщение <CR> <LF>              --- status line
название заголовка : значение <CR> <LF>             --- header line
.......                                             --- header line
<CR> <LF>
entity body                                         --- пересылаемые данные
```

Здесь полная аналогия с запросом с точностью до первой строчки. Например, ответ на предыдузий запрос мог выглядеть как-то так

```
HTTP/1.1  200 OK <CR> <LF>                                  --- все в порядке
Connection : close <CR> <LF>                                --- мы закрыли сооединение
Date : Wed, 05 Apr 2024  06:32:25 GMT <CR> <LF>             --- когда ответили
Server : Apache/1.3.0 (Unix) <CR> <LF>                      --- кто ответил
Last-modified : Mon, 27 Jan 2010  11:12:42 GMT <CR> <LF>--- когда изменялся этот файл
Content-length : 27631 <CR> <LF>                            --- длина файла в байтах
Content-type : text/html <CR> <LF>                          --- что в файле
<CR> <LF>
<содержимое файла file.html>
```

HTTP, коды отклика

Статусная строка ответа содержит версию протокола и код отклика с его текстовой расшифровкой Протокол определяет 5 категорий откликов, которые различатся первой цифрой кода

1xx — information
2xx — success
3xx — redirection
4xx — client error
5xx — server error

Вот некоторые примеры откликов

200 OK
301 Moved permanently
400 Bad request
404 Not found
505 HTTP version not supported

Соответственно клиент должен проанализировать этот код и далее поступать в соответствеии с логикой своей работы.

Здесь мы более не будем погружаться в глубины HTTP.