CGI-сценарии

Главным достижением технологии World Wide Web по праву считают унификацию интерфейса пользователя при работе с информационными ресурсами Internet. Универсальный мультипротокольный браузер, будь то Netscape Navigator или Internet Explorer, позволяет путем выбора гипертекстовой ссылки получить доступ к FTP-архиву, архиву Gopher, новостям из конференции Usenet или отправить письмо по электронной почте. До эпохи Web для каждого из этих ресурсов пришлось бы запускать отдельную программу.

Однако, кроме текстов, которые можно читать, или картинок, которые можно просматривать, существует множество ресурсов, требующих ввода информации в процессе работы с ними. К таким ресурсам, в частности, относятся информационно-поисковые системы, где пользователь должен вводить список ключевых слов или реляционные (да и любые другие) базы данных, формулируя запрос к отношениям. Более того, для любой страницы, которая требует аутентификации пользователя, необходимо вводить идентификатор и пароль.

Форматирование страниц в Web-технологии достигается за счет HTML-разметки. Остается только создать инструмент ввода данных через рабочее окно браузера или через HTML-документ. В 1991 году эта проблема была решена специалистами NCSA. Они разработали и реализовали две взаимосвязанные спецификации: HTML-формы и Common Gateway Interface.

Формы произвели настоящую революцию в HTML-разметке: авторы документов получили возможность создавать сложные шаблоны ввода информации в рамках HTML-страницы, пользователи — эти шаблоны заполнять. При этом авторы форм опирались на свойства HTTP-протокола и универсальный локатор ресурсов URL с учетом того, что при HTTP-обмене можно использовать различные методы доступа к ресурсам. Это позволило сделать механизм интерпретации форм расширяемым и легко приспосабливаемым к дальнейшему развитию Web-технологии. Таким образом, кроме HTTP, можно было использовать и другие протоколы, которые поддерживали универсальный браузер, например mailto.

Common Gateway Interface — это спецификация обмена данными между прикладной программой, выполняемой по запросу пользователя, и HTTP-сервером, который данную программу запускает. До появления CGI новые функции нужно было внедрять непосредственно в сервер. CGI позволила разрабатывать программы независимо от сервера, а механизм передачи им управления и данных был унаследован от программирования в среде командной строки. Последнее резко сократило трудозатраты на разработку приложений, так как не надо было программировать интерфейс пользователя: его функции выполняли формы.

В CGI имеет смысл выделить следующие основные моменты:

- понятие CGI-скрипта;
- типы запросов;
- механизмы приема данных скриптом;
- механизм генерации отклика скриптом.

Основное назначение CGI — обработка данных из HTML-форм. В настоящее время область применения CGI гораздо шире.

Понятие CGI-скрипта

CGI-скриптом называют программу, написанную на любом языке программирования или командном языке, которая осуществляет обмен данными с HTTP-сервером в соответствии со спецификацией Common Gateway Interface.

Наиболее популярными языками для разработки скриптов являются Perl и C.

HyperText Transfer Protocol

Все данные в рамках Web-технологии передаются по протоколу HTTP. Исключение составляет обмен с использованием программирования на Java или обмен из Plugin-приложений. Учитывая реальный объем трафика, который передается в рамках Web-обмена по HTTP, мы будем рассматривать только этот протокол. При этом мы остановимся на таких вопросах, как:

общая структура сообщений;
методы доступа;
оптимизация обменов.

Общая структура сообщений

HTTP — это протокол прикладного уровня. Он ориентирован на модель обмена "клиент-сервер". Клиент и сервер обмениваются фрагментами данных, которые называются HTTP-сообщениями. Сообщения, отправляемые клиентом серверу, называют запросами, а сообщения, отправляемые сервером клиенту — откликами. Сообщение может состоять из двух частей: заголовка и тела. Тело от заголовка отделяется пустой строкой.

Заголовок содержит служебную информацию, необходимую для обработки тела сообщения или управления обменом. Заголовок состоит из директив заголовка, которые обычно записываются каждая на новой строке.

Тело сообщения не является обязательным, в отличие от заголовка сообщения. Оно может содержать текст, графику, аудио- или видеоинформацию.

Ниже приведен HTTP-запрос:

GET / HTTP/1.0
Accept: image/jpeg
пустая строка
И отклик:

HTTP/1.0 200 OK
Date: Fri, 24 Jul 1998 21:30:51 GMT
Server: Apache/1.2.5
Content-type: text/html
Content-length: 21345
пустая строка
<HTML>
...
</HTML>
Текст "пустая строка" — это просто обозначение наличия пустой строки, которая отделяет заголовок HTTP-сообщения от его тела.

Сервер, принимая запрос от клиента, часть информации заголовка HTTP-запроса преобразует в переменные окружения, которые доступны для анализа CGI-скриптом. Если запрос имеет тело, то оно становится доступным скрипту через поток стандартного ввода.

Методы доступа
Самой главной директивой HTTP-запроса является метод доступа. Он указывается первым словом в первой строке запроса. В нашем примере это GET. Различают четыре основных метода доступа:

GET;
HEAD;
POST;
PUT.

Метод GET
Метод GET применяется клиентом при запросе к серверу по умолчанию. В этом случае клиент сообщает адрес ресурса (URL), который он хочет получить, версию протокола HTTP, поддерживаемые им MIME-типы документов, версию и название клиентского программного обеспечения. Все эти параметры указываются в заголовке HTTP-запроса. Тело в запросе не передается.

В ответ сервер сообщает версию HTTP-протокола, код возврата, тип содержания тела сообщения, размер тела сообщения и ряд других необязательных директив HTTP-заголовка. Сам ресурс, обычно HTML-страница, передается в теле отклика.

Метод HEAD
Метод HEAD используется для уменьшения обменов при работе по протоколу HTTP. Он аналогичен методу GET за исключением того, что в отклике тело сообщения не передается. Данный метод используется для проверки времени последней модификации ресурса и срока годности кэшированных ресурсов, а также при использовании программ сканирования ресурсов World Wide Web. Одним словом, метод HEAD предназначен для уменьшения объема передаваемой по сети информации в рамках HTTP-обмена.

Метод POST
Метод POST — это альтернатива методу GET. При обмене данными по методу POST в запросе клиента присутствует тело HTTP-сообщения. Это тело может формироваться из данных, которые вводятся в HTML-форме, или из присоединенного внешнего файла. В отклике, как правило, присутствует и заголовок, и тело HTTP-сообщения. Чтобы инициировать обмен по методу POST, в атрибуте METHOD контейнера FORM следует указать значение "post".

Метод PUT
Метод PUT используется для публикации HTML-страниц в каталоге HTTP-сервера. При передаче данных от клиента к серверу в сообщении присутствует и заголовок сообщения, в котором указан URL данного ресурса, и тело — содержание размещаемого ресурса.

В отклике тело ресурса обычно не передается, а в заголовке сообщения указывается код возврата, который определяет успешное или неуспешное размещение ресурса.

Оптимизация обменов
Протокол HTTP изначально не был ориентирован на постоянное соединение. Это означает, что как только сервер принял запрос от клиента и ответил на него, соединение между

клиентом и сервером разрывается. Для нового обмена данными нужно устанавливать новое соединение. Такой подход имеет как достоинства, так и недостатки.

К достоинствам относится возможность одновременного обслуживания большого количества коротких запросов. Даже на популярных серверах число открытых соединений может не превышать сотни при обслуживании порядка миллиона запросов в сутки. При этом один клиент может открыть до 40 соединений одновременно, и с точки зрения сервера все они равноправны. При высокоскоростных линиях связи это позволяет добиться малого времени отклика на запрос клиента для всей страницы (текст, графика и т.п.).

К недостаткам такой схемы обмена относятся: необходимость каждый раз устанавливать соединение и невозможность поддерживать сессию работы с информационным ресурсом. При инициализации соединения по транспортному протоколу TCP и разрыве этого соединения требуется передать довольно большой объем служебной информации. Отсутствие поддержки сессий в HTTP затрудняет работу с такими ресурсами как базы данных или ресурсы, требующие аутентификации.

Для оптимизации числа открытых TCP-соединений в HTTP-протоколе версий 1.0 и 1.1 предусмотрен режим keep-alive. В этом режиме соединение инициализируется только один раз, и по нему последовательно можно реализовать несколько HTTP-обменов.

Для обеспечения поддержки сессий к директивам HTTP-заголовка были добавлены "ключики" (cookies). Они позволяют сымитировать поддержку соединения при работе по протоколу HTTP.

Виды интерфейса пользователя в Web-технологии
Страницы World Wide Web по функциональному назначению можно разделить на несколько типов: информационные страницы, навигационные страницы, страницы обмена данными. Во многих случаях эти функции можно объединить в одной странице.

Информационные страницы — это последовательное изложение информации с возможностью гипертекстовых контекстных переходов. Пользователь просматривает их последовательно. Гипертекстовые ссылки обычно применяют для создания сносок, примечаний или отсылок к спискам литературы и других ассоциативных материалов. Типичными примерами таких страниц являются подсказки, руководства, описания компаний, исторические справки и т.п.

Навигационные страницы — это совокупность гипертекстовых ссылок, которая позволяет ориентироваться в материалах Web-узла. Типичный пример такой страницы — Home page (домашняя страница). Как правило, на ней нет пространных текстовых описаний и иллюстраций, она состоит из совокупности различных меню. Эти меню можно реализовать через списки, таблицы ссылок или imagemap.

Страницы обмена данными позволяют передать на сервер некоторый объем информации, отличный от стандартного адреса (URL) ресурса. При просмотре и навигации пользователь просто выбирает гипертекстовые ссылки, по которым загружаются новые страницы. При обмене данными на сервер передается не только адрес ресурса, но и дополнительная информация, которую вводит пользователь.

В зависимости от функционального назначения страниц изменяется вид интерфейса ресурса, с которым пользователь имеет дело. В первых двух случаях достаточно манипулятором "мышь" выбрать гипертекстовую ссылку, как тут же загрузится новая страница. В случае

страниц обмена данными следует заполнить поля HTML-форм и отправить данные на сервер.

При этом формы обеспечивают практически все необходимые виды полей ввода и меню. Единственное, чего не позволяют реализовать HTML-формы, так это вложенные меню. Формы можно применять не только при обмене данными. Достаточно развитые механизмы обработки форм присутствуют в JavaScript.

Механизмы приема данных скриптом
Скрипт может принять данные от сервера тремя способами:

через переменные окружения;
через аргументы командной строки;
через поток стандартного ввода.
При описании этих механизмов будем считать, что речь идет об обмене данными с сервером Apache для платформы Unix.

Переменные окружения
При вызове скрипта сервер выполняет системные вызовы fork и exec. При этом он создает среду выполнения скрипта, определяя ее переменные. В спецификации CGI определены 22 переменные окружения. При обращении к скрипту разными методами и из различных контекстов реальные значения принимают разные совокупности этих переменных. Например, при обращении по методу POST переменная QUERY_STRING не имеет значения, а по методу GET — имеет. Другой пример — переменная окружения HTTP_REFERER. При переходе по гипертекстовой ссылке она определена, а если перейти по значению поля location или через JavaScript-программу, то HTTP_REFERER определена не будет.

Получить доступ к переменным окружения можно в зависимости от языка программирования следующим образом:

```
#Perl
$a = $ENV{CONTENT_LENGTH};
...
// C
a = getenv("CONTENT_LENGTH");
```

В случае доступа к скрипту по методу GET данные, которые передаются скрипту, размещаются в переменной окружения QUERY_STRING.

Аргументы командной строки
Как ни странно звучит, но у CGI-скрипта может быть такой элемент операционного окружения как командная строка. Это не означает, что скрипт реально можно вызвать из командной строки через сервер. Тем не менее получить доступ к содержанию командной строки скрипта можно с помощью тех же функций, что и при вызове его из-под интерактивной оболочки:

```
#Perl
foreach $a (@ARGV)
{
print $a,"\n";
}
```

```
// C
void main(argc,argv)
int argc;
char *argv[];
{
int i;
for(i=0;i<argc;i++)
{
printf("%s\n",argv[i]);
}
}
```

В обоих примерах показана распечатка аргументов командной строки для программ на Perl и C соответственно.

Аргументы командной строки появляются только в запросах типа ISINDEX.

Поток стандартного ввода

Ввод данных в скрипт через поток стандартного ввода осуществляется только при использовании метода доступа к ресурсу (скрипту) POST. При этом в переменную окружения CONTENT_LENGTH помещается число символов, которое необходимо считать из потока стандартного ввода скрипта, а в переменную окружения CONTENT_TYPE помещается тип кодирования данных, которые считываются из потока стандартного ввода.

При посимвольном считывании в C можно применить, например, такой фрагмент кода:

```
int n;
char *buf;
n= atoi(getenv("CONTENT_LENGTH"));
buf = (char *) malloc(n+1);
memset(buf,'\000',n+1);
for(i=0;i<n;i++)
{
buf[i]=getchar();
}

free(buf);
```

В данном фрагменте применено динамическое размещение памяти в скрипте, поэтому при выходе из него память следует освободить. Вообще говоря, память будет автоматически освобождена операционной системой после завершения скрипта. Однако, если переносить скрипт на спецификацию FCGI (Fast CGI), что требует минимума переделок, из-за неаккуратной работы с памятью могут возникнуть проблемы.

Механизм генерации отклика скриптом

Существует только один способ вернуть данные серверу и, соответственно, браузеру пользователя — писать в поток стандартного вывода (STDOUT). При этом скрипт должен формировать HTTP-сообщение.

Сначала выводятся директивы HTTP-заголовка. В минимальном варианте это либо

```
Content-type: text/html,
```

либо

Location: http://intuit.ru/
В первом случае определяется тип тела HTTP-сообщения, а во втором осуществляется перенаправление запроса.

После заголовка генерируется отклик в виде тела HTTP-сообщения, которое должно быть отделено от заголовка пустой строкой:

```
#!/bin/sh
echo Content-type: text/plain
echo
echo Hello
```

В данном случае используется командный интерпретатор sh.

Если скрипт начинает формирование заголовка с директивы версии HTTP-протокола, то сервер не анализирует отклик и передает его как есть. Если в заголовке, сгенерированном скриптом, эта директива отсутствует, то сервер считает, что заголовок неполный, и вставляет в него дополнительные директивы.