Веб-приложения

В первые годы развития World Wide Web (WWW) большинство веб-сайтов полностью строилось на основе статических HTML-страниц, содержимое которых нельзя было изменить путем обращения к ним. По мере развития веб-технологий сайты стали включать различные способы создания и изменения страниц в ответ на действия пользователя, которые называются динамическими страницами.

Веб-приложение – это приложение, доступное через WWW, или, иначе говоря, предоставляющее веб-интерфейс (то есть интерфейс, описываемый на языке HTML). Основным достоинством веб-приложений является простота доступа к приложению, так как для этого пользователю необходимо иметь только браузер. Кроме того, значительно упрощается сопровождение приложения. Функционирование веб-приложения неразрывно связано с веб-сервером – системным ПО, обеспечивающим прием ввода пользователя и вывода результатов в виде сообщений протокола передачи гипертекста (Hypertext Transfer Protocol – HTTP).

Первые веб-приложения строились на базе интерфейса Common Gateway Interface (CGI), являющегося стандартом связи внешней прикладной программы с веб-сервером. С помощью CGI можно создавать CGI-программы, называемые шлюзами, которые во взаимодействии с такими прикладными системами, как система управления базой данных, электронная таблица, и др., смогут выдать на экран пользователя динамическую информацию. Программа-шлюз запускается веб-сервером в реальном масштабе времени. Для передачи данных об информационном запросе от веб-сервера к шлюзу используется командную строку и переменные окружения. Эти переменные окружения устанавливаются в тот момент, когда веб-сервер выполняет программу шлюза. Шлюз осуществляет свой вывод в стандартный поток вывода. Этот вывод может представлять собой или документ, сгенерированный шлюзом, или инструкции серверу, где получить необходимый документ. Программа-шлюз может быть закодирована на языках C/C++, Fortran и т.п., а также на скриптовых языках, таких как Perl, TCL, Unix Schell и т.п.. Программы-шлюзы обычно размещались в подкаталоге `cgi-bin` веб-сервера. Недостатком веб-приложений на базе CGI является невысокая производительность и плохая масштабируемость, так как для обработки каждого запроса создается отдельный процесс ОС, выполняющий программу-шлюз.

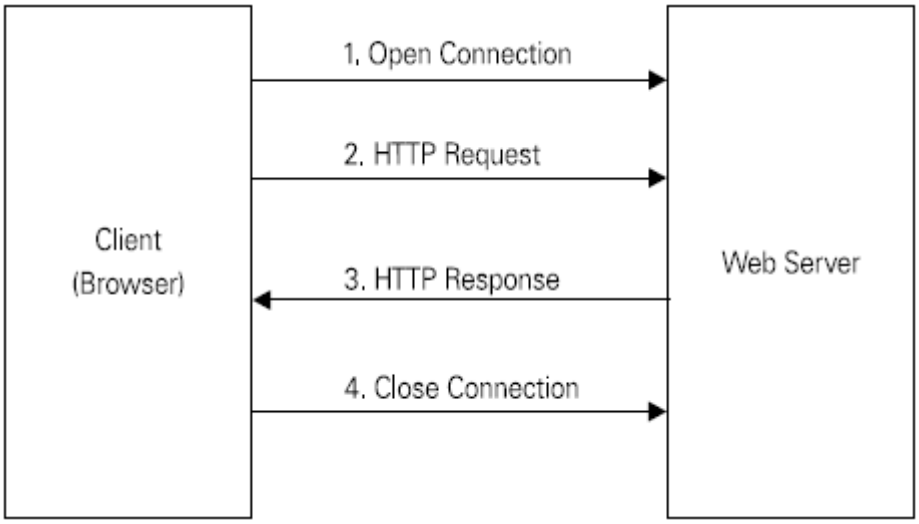
Следующим этапом развития веб-приложений стало появление серверов веб-приложений, которые ориентируются на эффективную генерацию динамических страниц и предоставление дополнительных функций для веб-приложений (управление жизненных циклом приложения, обеспечение безопасности, организация пулов ресурсов, в том числе пулов подключений к источникам данных и т.д.). Как правило, сервер веб-приложений предусматривает достаточно тесную интеграцию с выполняемыми в нем приложениями, поэтому веб-приложения для него разрабатываются в виде подключаемых модулей, не предполагающих самостоятельное использование и взаимодействующих с сервером через тот или иной интерфейс (API). Так, веб-приложения для Microsoft Internet Information Server реализуются в виде DLL и взаимодействуют с сервером через интерфейс ISAPI. Веб-приложения на платформе Java EE распространяются в виде WAR-архивов и взаимодействуют с веб-контейнером через Servlet API.

Взаимодействие по протоколу HTTP

Протокол HTTP – это протокол прикладного уровня для распределенных гипермедийных информационных систем. HTTP используется в WWW, начиная с 1990 года. Первой версией HTTP, известной как HTTP/0.9, был простой протокол для передачи необработанных данных через Интернет. В следующей версии протокола – HTTP/1.0 – был определен MIME-подобный формат сообщений, содержащий метаинформацию о передаваемых данных и модификаторы семантики запросов/ответов. Однако HTTP/1.0

недостаточно хорошо учитывал особенности работы с иерархическими прокси-серверами, кэшированием, постоянными соединениями, и виртуальными хостами. Кроме того, быстрое увеличение количества не полностью совместимых приложений, использующих протокол HTTP/1.0, потребовало введения версии протокола, в которой были бы заложены возможности, позволяющие приложениям определять истинные возможности друг друга. Протокол HTTP/1.1 содержит более строгие требования, чем HTTP/1.0, гарантирующие надежную реализацию возможностей.

Реальным информационным системам требуются большая функциональность, чем просто загрузка информации, в том числе поиск, модификация и аннотирование ресурсов. Протокол HTTP предоставляет расширяемый набор методов и заголовков, которые указывают цель запроса. Для идентификации ресурса, к которому следует применить метод, в протоколе HTTP используются универсальные идентификаторы ресурсов (URI).



Протокол HTTP – это протокол типа запрос/ответ. Клиент устанавливает соединение с сервером и посылает по нему запрос, а сервер в результате выполнения запрашиваемого действия посылает клиенту ответ, после чего клиент закрывает соединение. Взаимодействие по протоколу HTTP обычно происходит посредством TCP/IP-соединений. По умолчанию используется порт TCP 80, но могут использоваться и другие порты. Большинство реализаций HTTP/1.0 использовало новое соединение для каждого обмена запросом/ответом. В HTTP/1.1 установленное соединение может использоваться для одного или нескольких обменов запросом/ответом, что позволяет значительно повысить эффективность взаимодействия.

Протокол HTTP – это протокол без состояния, то есть ответ на запрос от определенного клиента не зависит от его предыдущих запросов. С точки зрения сервера любой запрос от клиента является первым.

Из сказанного выше ясно, что HTTP-сообщения делятся на запросы клиента серверу и ответы сервера клиенту. И те, и другие используют обобщенный формат сообщения RFC 822 для пересылки объектов (полезной нагрузки сообщения). Оба типа сообщений выглядят следующим образом: сначала идет начальная строка (`start-line`), затем один или несколько полей заголовка (называемых также просто "заголовки"), затем пустая строка, указывающая конец полей заголовка, а затем, возможно, тело сообщения.

```
generic-message = start-line
                  *message-header
                  CRLF
                  [ message-body ]
```

Поля заголовков HTTP, которые включают поля общих заголовков, заголовков запроса, заголовков ответа, и заголовков объекта, имеют обобщенный формат, описанный в RFC 822. Каждое поле заголовка состоит из имени, двоеточия (":") и значения поля. В сообщении могут присутствовать несколько полей заголовка с одинаковыми именами, если их значения могут образовывать разделенный запятыми список.

Тело HTTP-сообщения (message-body), если оно присутствует, используется для передачи тела объекта, связанного с запросом или ответом. Тело сообщения отличается от тела объекта (entity-body) только в том случае, когда применяется кодирование передачи, что указывается полем заголовка Transfer-Encoding.

```
message-body = entity-body
             | <entity-body, закодированное согласно Transfer-Encoding>
```

Тело объекта (если оно присутствует) имеет формат и кодирование, определяемое полями заголовка объекта Content-Type и Content-Encoding. соответственно. Эти заголовки определяют двухуровневую упорядоченную модель кодирования:

```
entity-body := Content-Encoding( Content-Type( data ) )
```

Тип содержимого (Content-Type) определяет MIME-тип объекта. Многие протоколы (HTTP, SMTP и др.) позволяют передавать информацию самых разных типов, требующих различной обработки. Поэтому появилась необходимость идентифицировать тип передаваемой информации. Это делается с помощью идентификатора MIME-типа – строки вида тип/подтип. Так, для текстовых документов установлены следующие MIME-типы: text/plain – только текст, text/html – HTML-документ, text/xml – XML-документ; для изображений: image/gif, image/jpeg, image/png; для видеоматериалов: video/mpeg и др.; для сложных документов: application/pdf – PDF-документ, application/x-msword – документ MS Word и др.

Кодирование содержимого (Content-Encoding) может использоваться для указания любого дополнительного кодирования содержимого, примененного к данным (обычно с целью сжатия данных).

Рассмотрим подробнее структуру HTTP-запроса.

Сообщение запроса от клиента к серверу содержит в первой строке метод, который нужно применить к ресурсу, универсальный идентификатор ресурса (URI) и используемую версию протокола.

Пример запроса:

```
GET /servlets-examples/servlet/RequestParamExample HTTP/1.1
Accept: image/gif, image/jpeg, application/msword, */*
Referer: http://localhost:8084/servlets-examples/
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2)
Host: localhost:8084
Connection: Keep-Alive
                                                          | ← пустая строка
```

Идентификаторы ресурсов (URI) делятся на имена (URN) и локаторы (URL), в HTTP-запросах используются преимущественно последние. В общем случае URL для ресурсов, доступных по протоколу HTTP, состоит из восьми частей:

```
schema://[user:password@]host[:port]/[path][?query][#hash]
```

, где schema – схема, определяющая правила кодирования URL и соответствующая определенному прикладному протоколу (для протокола HTTP допустимы значения http и https); user и password – имя пользователя и пароль для доступа к ресурсу; host – полностью квалифицированное имя домена или IP-адрес сервера; port – порт сервера (для HTTP по умолчанию 80); path – логический путь и имя ресурса на сервере; query – параметры запроса; hash – идентификатор местоположения в документе-ответе, до которого браузер прокрутит его по получении.

В протоколе HTTP предусмотрено семь видов запросов (методов), из которых важнейшими с точки зрения организации интерактивного взаимодействия с клиентом являются методы GET и POST.

Метод	Описание
OPTIONS	Запрос информации о коммуникационных возможностях сервера.
GET	Получение информации с сервера.
HEAD	Получение заголовков. Идентичен методу GET, но ответ не должен

	содержать тела сообщения, однако все заголовки должны быть установлены (в том числе заголовки объекта, такие как `Content-Type`, `Content-Length` и прочие).
POST	Обработка содержащегося в запросе объекта как подчиненного относительно запрашиваемого ресурса.
PUT	Сохранение содержащегося в запросе объекта по указанному адресу.
DELETE	Удаление указанного в запросе ресурса.
TRACE	Трассировка запроса – возвращение текста запроса в теле ответа.

Теперь рассмотрим структуру HTTP-ответа. Первая строка ответа – это строка статуса, содержащая версию протокола, кода статуса и поясняющую фразу.

Код статуса – это целочисленный трехразрядный код результата выполнения запроса. Первая цифра кода статуса определяет класс ответа:

- 1xx: Информационные коды - запрос получен, продолжается обработка.
- 2xx: Успешные коды - действие было успешно получено, понято и обработано.
- 3xx: Коды перенаправления - для выполнения запроса должны быть предприняты дальнейшие действия.
- 4xx: Коды ошибок клиента - запрос имеет плохой синтаксис или не может быть выполнен.
- 5xx: Коды ошибок сервера - сервер не в состоянии выполнить допустимый запрос.

Приведем наиболее распространенные коды статуса:

- 200 – OK;
- 301 – Moved Permanently (Перемещен навсегда);
- 302 – Moved Temporarily (Временно перемещен);
- 403 – Forbidden (Доступ запрещен);
- 404 – Not Found (Ресурс не найден);
- 500 – Internal Server Error (Внутренняя ошибка сервера).

Пример ответа:

```
HTTP/1.1 200 OK
Date: Tue, 01 Dec 2001 23:59:59 GMT
Content-Type: text/html
Content-Length: 51
                                                    | ← пустая строка
<html>
<body>
<h1>Hello, John!</h1>
</body>
</html>
```

Организация диалога с пользователем

Для организации диалога веб-приложения с пользователем непосредственно в браузере в языке HTML предназначены формы (элемент `form`). Например, для аутентификации пользователя можно использовать следующую форму:

```
<form name="login" action="controller" method="post" enctype="MIME-тип">
Username: <input name="username" type="text"></p>
Password: <input name="password" type="password"></p>
<input name="action" type="hidden" value="login">
<input type="submit" value="Войти">
</form>
```

Поскольку результаты ввода данных пользователем в форму должны быть переданы на сервер и обработаны, с формой необходимо связать URL программы-обработчика, который указывается обязательным атрибутом `action`.

Метод передачи данных формы указывается атрибутом `method`. Определено два метода: `get` и `post`, соответствующих в HTTP-запросах методам GET и POST. В первом

случае данные формы передаются как часть URL серверной программы в виде последовательности пар `имя_элемента_формы=значение`, разделенных символом амперсанда (`&`). При этом недопустимые в URL символы (например, пробелы, спецсимволы) кодируются в виде %XX, где XX - шестнадцатеричный код символа (так, пробел будет заменен на %20). Поскольку некоторые системы накладывают ограничения на длину URL (255 символов), использовать этот метод не рекомендуется, так как объем данных формы может быть достаточно большим. Пример URL для приведенной выше формы, если бы метод передачи данных был `get`:

```
http://www.vstore.com/controller?username=test&password=test&action=login
```

Для форм больше подходит метод POST, поскольку в этом случае данные формы передаются в теле HTTP-сообщения, которое может содержать большой объем информации. По умолчанию используется метод `get`, поэтому метод `post` нужно указывать явно.

Атрибут `enctype` определяет формат кодирования данных формы при передаче их методом POST. Для данного атрибута определено два значения: `application/x-www-form-ulrencoded` и `multipart/form-data`. Первое используется по умолчанию, второе необходимо при передаче на сервер файлов (используется в сочетании с элементом управления типа `file`, см. ниже) и применяется в специфических случаях, например, при передаче вложений почтовому серверу через Web-интерфейс. Пример HTTP-запроса при использовании формата `application/x-www-form-ulrencoded`:

```
POST /controller HTTP/1.1
Host: www.vstore.com
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2)
Content-Type: application/x-www-form-ulrencoded
Content-Length: 40
                                                    | ← пустая строка
username=test
password=test
action=login
```

Язык HTML определяет набор элементов для описания элементов управления, которые можно использовать для конструирования форм. Эти элементы указывают внутри элемента `form`. Элементы управления формы описываются HTML-элементами `input`, `select` и `textarea`. Два последних представляют собой список опций и многострочную область редактирования. HTML-элемент `input` предназначен для создания элемента управления, конкретный тип которого указывается атрибутом `type`:

- `checkbox` – переключатель с двумя состояниями: отмечено/не отмечено;
- `button` – обычная кнопка;
- `file` – элемент выбора локального файла;
- `hidden` – скрытый элемент формы, не отображается на экране, используется только для передачи данных;
- `password` – строка редактирования для ввода пароля, ввод при отображении заменяется символами `*`;
- `radio` – радио-кнопка;
- `reset` - кнопка восстановления исходных значений элементов управления формы;
- `submit` - кнопка передачи данные формы на сервер;
- `text` - строка редактирования, используется по умолчанию.

Параметры запроса формируются на основе элементов `input`, `select` и `textarea`, описанных внутри элемента `form`. Имя параметра соответствует атрибуту `name` элемента `input`, а значение – либо атрибуту `value`, явно указанному для элемента `input`, либо значению, введенному пользователем в соответствующий элемент управления. Порядок определения значения зависит от типа элемента управления.