Информатика

http-серверы: продолжение

© Марченко Антон Александрович 2017 г.
Абрамский Михаил Михайлович

Ранее…

- HttpListener
- HttpListenerContext
- HttpListenerRequest
- HttpListenerResponse
- HttpListenerResponse.OutputStream

Научились

- Обрабатывать входящие запросы «на лету»
  - GET, POST, …
- Динамически генерировать ответы
- Работать с host name, port, path, parameters

Но…

- Все запросы обрабатывались однотипно
- ***Что, если обработка нескольких запросов подряд должна отличаться от обработки каждого запроса по отдельности?***

HTTP-*stateless*

- Каждый запрос должен обрабатываться отдельно от других, не завися от предыдущих запросов
- **Но всегда ли сайты работают так?**
  - Всегда ли GET запрос к vk.com/ с одного клиента возвращает одинаковое содержимое?

Очевидно, нет

- **В случае vk.com**
  - GET vk.com/ при первом обращении ведёт на страницу логина
  - GET vk.com/ после входа на сайт ведёт на страницу новостей

Значит, на сервере хранится информация, влияющая на обработку запросов

  - по одному request'у могут генерироваться разные response'ы

Вопросы

- Что это за информация?
- Как хранится на сервере?
- Общая для приложения?
- Или своя для каждого исполняемого компонента?
- Или ещё как-то?

Чего у сервера *несколько*?

- У одного приложения – ***несколько*** *обработчиков (handler),* возможно **исполняемых компонент** (процессов, EXE)
  - Google: веб-ферма, веб-сад
- Каждая компонента может обрабатывать *несколько* запросов

Пример:

«100500 пользователей сейчас на сайте»

- 100500 пользователей обработались компонентой AuthServer, о чем можно было сохранить информацию
  - ***где можно сохранить эту информацию?***

Состояние обработчика

- Можно сохранять данные в ассоциативную коллекцию (словарь)
  - с уникальными строковыми ключами
- Так работает ViewState в ASP.NET WebForms

Скрытые поля страницы

- Можно сохранять специфичные для страницы данные о состоянии в скрытом поле на самой странице
  - Просто, тупо, небезопасно
- Также приходится вместо GET запросов использовать POST и отправлять страницу на сервер

Очевидно

У каждой компоненты есть свои собственные данные, которые она использует при обработке запроса

Достаточно ли такого разделения?

А что делать, если…

- «Вы уже проголосовали сегодня»
- «В вашей корзине 2 товара»
- «В ваш аккаунт выполнен вход с другого устройства»
- «У вас нет прав для просмотра этой страницы. Пожалуйста, авторизуйтесь»

Есть зависимость от конкретного клиента!
(а данные могут быть общими для разных компонент)

Чего у сервера *несколько*?

- У одного приложения – *несколько* клиентов
  - все пользуемся одним сайтом
- Запросы *одного* клиента обрабатывают *несколько* компонент
  - Авторизуемся, смотрим новости, переписываемся, слушаем музыку, это разные запросы для разных обработчиков
  - **Нужно общее хранилище данных, общее для обработчиков, индивидуальное для каждого клиента**

Состояние сессии

- Данные, сохраняемые на сервере пока пользователь работает с веб-приложением
  - выполняет несколько запросов
- Обычно – словарь, хранящийся в памяти сервера
  - например, Google: session state ASP.NET Core
- Время хранения ограничено

Хранение сессий

- Может стать весельем в случае веб-фермы
- Типичные подходы к хранению сессий:
  - Локально
  - В отдельной службе
  - В базе данных
- **В чем плюсы и минусы подходов?**

Пользователи

- С помощью сессии можно проверить, был ли авторизован пользователь
- Можно в словаре сессии проверить наличие значения по ключу (например, currentUser)
  - В зависимости от результата направить к обработке или к форме авторизации

Кстати

- **Идентификация** – представление пользователя, сообщение о том, кто он такой
- **Аутентификация** – проверка доказательств (факторов) подтверждения личности
  - Идентификация – часть аутентификации
- **Авторизация** – проверка разрешений действий аутентифицированной персоне

Хабр + еще ссылка

Перенаправление

- **Redirect** – автоматический переход на другую страницу, инициируемый сервером
- Содержимое адресной строки меняется без участия пользователя
- **Происходит на любом сайте**
  - пример с перенаправлением анонимных и авторизованных пользователей

Перенаправление

У HttpListenerResponse

```
context.Response.Redirect("http://ya.ru/");
```

Клиенту возвращается response с кодом 302, в заголовках содержится **Location** с адресом перехода

Wait!

- У сервера много клиентов
- **Как сервер определяет клиента, работая именно с его сессией?**
  - “Вычисляет по IP”?

Не по IP

- У одного пользователя много браузеров (клиентов)
- Для каждого браузера (клиента) своя сессия
- У браузера есть то, что позволяет серверу узнавать его

Печенька

Cookie

- Пара строк «ключ-значение», сохранённая в браузере для конкретного ресурса
- Посылается с любым запросом к этому ресурсу

Сервер может…

- Обрабатывать запрос, учитывая пришедшие cookies

- Добавлять к ответу новые cookies или изменять существующие

Response header Set-Cookie

Анонимный вход

twitter.com

Set-Cookie:

guest_id=v1%3A144278780784898471; Domain=.twitter.com; Path=/; Expires=Tue, 19-Sep-2017 22:23:27 UTC

Set-Cookie:

_twitter_sess=BAh7CSIKZmxhc2hJQzonQWN0aW9uQ29udHJvbGxlcjo6Rmxhc2g6OkZsYXNo%250ASGFzaHsABjoKQHVzZWR7ADoPY3JlYXRlZF9hdGwrCJjS2uxPAToMY3NyZl9p%250AZCIlZWVmZjZiMjljYmNhODQ0ZTAwZTM5ZTczYzBiYjAxODg6B2lkIiU2NDY5%250A……..--e08bcf728ab470376430b7ca34de6b30c84792d7; Path=/; Domain=.twitter.com; Secure; HTTPOnly

vk.com

Set-Cookie:

remixlang=0; expires=Mon, 26 Sep 2016 00:38:03 GMT; path=/; domain=.vk.com

Set-Cookie:

remixlhk=94d62adefb7d0ed9e0; expires=Sat, 10 Sep 2016 05:16:22 GMT; path=/; domain=.vk.com

Request header Cookie

А теперь мы авторизованы (обычный режим браузера)

twitter.com

Cookie:

guest_id=v1%3A142099846466837593; eu_cn=1; remember_checked_on=1;
device_token="Oo+Qpr2ZUvnqFDjw....."; auth_token=f76f24e……..; webn=164339606;
external_referer=padhuUp37zhVH6NUFmkhcHEHTK7ZZOKn|0;
h=%5B%7B%22newer_tweet_id%22%3A%2264572412414538956 8%22%2C%22older_tweet_id%22%3A%2264572310986625843 2%22%2C%22promoted_content%22%3A%7B%22impression_id%22%3A%2276711f3f359ca27f%22%2C%22disclosure_type%22%3A%22promoted%22%2C%22disclosure_text%22%3A%22%22%7D%2C%22experiment_values%22%3A%7B%22website_card_variation%22%3A%220%22%7D%2C%22advertiser_id%22%3A%221515423703%22%2C%22created_at%22%3A%221442788407%22%2C%22tweet_id%22%3A%226440555287761633 28%22%7D%5D; lang=ru; _gat=1; _ga=GA1.2.8335280.1421870119; dnt=1;
_twitter_sess=BAh7CSIKZmxhc2hJQzonQWN0aW9uQ29udHJvbGxlcjo6Rmxhc2g6Ok …… - 52160788eec4f951fb6a5cc0cffb314a1da66331

vk.com

Cookie:

remixtst=f17e9210; remixdt=0; …. ; remixlang=0; remixfeed=*.*.*.*.*.*.pr.*.*; audio_

У Cookie есть

- Имя, значение (строки)
- Expires – Время действия (GMT time)
  - `var cookieDate = DateTime.UtcNow.ToString("dddd, dd-MM-yyyy hh:mm:ss GMT");`
- Domain, Path – область действия (к какой части доменного имени и path применяется)

Cookie в http-сервере

```
var cookies=context.Request.Cookies;
foreach(Cookie cookie in cookies)
{
    ...
}

var cookie = new Cookie("uid", "123");
var expires=DateTime.UtcNow.AddMinutes(20)
            .ToString("dddd, dd-MM-yyyy hh:mm:ss GMT");
context.Response.Headers
       .Add("Set-Cookie",
       "cookieName=cookieValue;Path=/;Expires="
       + expires);
```

Web-приложение

- Клиент получает при доступе к серверу идентификатор сессии
- С его помощью сервер узнаёт клиента, работает с его сессией
- Если запретить cookie в браузере, то всё это работать не будет
  - можно исхитриться и передавать данные в URL запроса (ASP.NET так умеет)

Слава печенькам!

- Это они сохраняются, когда отмечаем галочку «запомнить меня»
- Сервер, приняв cookie, проводит автоматическую авторизацию пользователя

Берём выше

- Умеем хранить данные в сессиях, умеем сохранять данные в обработчиках/компонентах/процессах
- Что, если нужен общий параметр на все компоненты и все сессии
  - Счётчик посещений всех страниц всеми пользователями!

Состояние приложения

ApplicationState

- Можно использовать синглтон и инъекцию зависимостей
  - См. как реализовано в ASP.NET Core

Dear http-server,

Let’s stay just friends.

It’s complicated…

Бонус

Middleware

- Сквозные компоненты, образующие *конвейер* между сервером и приложением для проверки, маршрутизации и ответа
- Request’ы передаются по конвейеру, каждый middleware выполняет свою задачу

Проблемы с генерацией ответа

- Подход «в лоб» – генерировать ответ в строковом виде прямо в исходном коде

- **Какие проблемы при этом возникают?**

Шаблонизаторы

- Использование html-шаблонов для генерации конечных html-страниц
- **Основная цель – отделить представления от исходного кода**

Шаблон

- Содержит:
  - Статические элементы (html, css, js)
  - Динамические элементы, генерирующие содержимое, связывающие страницу с визуализируемыми данными

Пример шаблонизатора

- Шаблонизатор **Razor** из ASP.NET MVC
  - Google: ASP.NET MVC Razor
  - Пример
- Есть решения по использованию его вне ASP.NET (ссылка на хабр)
- В других языках:
  - ActionView в Ruby on Rails
  - DjangoTemplates (Python), Go Templates (Golang)
  - JSP в Java

Вопросы?

e-mail: marchenko@it.kfu.ru