ПИ,ПОИТ-4

Лекция 03

**Общие принципы построения web-приложений**

1. **Общие принципы построения web-приложений:**

* web-ресурсы приложения;
* запросы и ответы;
* соединение(сессия);
* конфигурационный файл приложения;
* контекст приложения;
* фильтры;
* кэш (данных и вывода);
* слушатели событий;
* принципы безопасности.

1. **Web-ресурс приложения:** сущность, расположенная на стороне сервера и имеющая URL/URI, к которой можно сделать http-запрос и получить http-ответ. Одно web-приложение представлено одним или более ресурсов.
2. **Web-ресурсы приложения: статические** - отправляются клиенту без изменения (html-страницы, рисунки, видео-файлы, …), **динамические** – динамически (программно) формируются на сервере и отправляются клиенту (сервлеты, JSP, http-обработчики, aspx-страницы,…). Ресурс может быть статическим относительно сервера и динамическим относительно клиента (html-страницы с JavaScript).

1. **Запрос(Request):** серверныйобъект, который образуется в результате обработки сервером http-запроса, поступающего от клиента и передается серверному программному коду для обработки. Содержит: всю информацию из http-запроса: метод, коллекция заголовков, коллекция параметров, поток данных … Обычно объект Request предоставляет возможность хранить данные в формате ключ/значение.

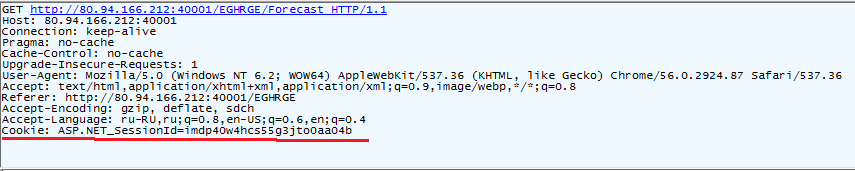
1. **Ответ(Response):** серверныйобъект, который автоматически формируется сервером, при получении http-запроса (одновременно с объектом Request), заполняется данными серверными программным кодом, преобразуется в http-ответ и отправляется клиенту. Содержит: всю информацию, которая должна быть помещена в http-ответ: статус, коллекция заголовков, поток данных, …

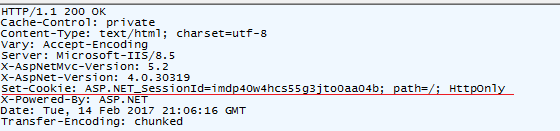


1. **Сессия (Session):** серверный объект, хранящий информацию о соединении с клиентом, создается при первом обращении время жизни: **timeout** (системный параметр, обычно равен 10 – 30 минутам) – максимальное время между запросами клиента. Если **timeout** превышен, то Session разрушается и при следующем запросе создается новый экземпляр. Каждая сессия имеет собственный идентификатор (**Session ID**, 16 или более байт). Каждый Requestпринадлежит, какой-то сессии (имеет ссылку на объект Session или содержит Session ID). Обычно объект Session предоставляет приложению возможность хранить данные в формате ключ/значение.

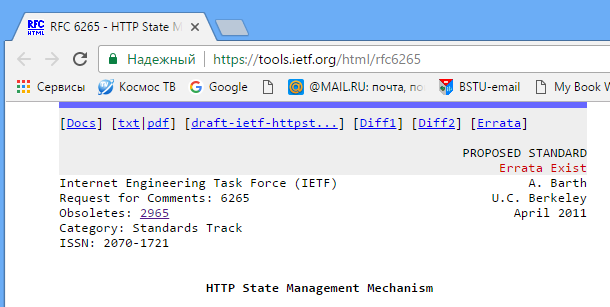


1. **Cookie**:фрагмент данных, оправленный web-сервером и хранимый web-клиентом. Используется для аутентификации, хранения пользовательских предпочтений, статистики, **информации о сеансе**  (обычно Session ID). Обычно имеет имя, содержащее URL, может иметь срок действия. Для создания и пересылки Cookie применяются заголовки.





1. **RFC 6265**: стандарт описывающий механизм Cookie.



1. **Конфигурационный файл web-приложения**: файл содержащий системные параметры приложения, обычно в XML-формате, служит основой для создания контекста web-приложения.
2. **Контекст web-приложения**: серверный объект, предназначенный для хранения информации об одом web-приложении, общий для всех файлов. Как правило, формируется сразу при загрузке web-сервера, основные данные (параметры приложения) копируются из конфигурационного файла приложения, общий для всех сессий приложения, Обычно контекст предоставляет возможность хранить данные в формате ключ/значение.



1. **Фильтр (Filter)**: серверный объект – препроцессор запроса, предназначен для предварительной обработки объекта Request. К одному ресурсу может быть построена цепочка фильтров, последний в цепочке – ресурс. Фильтр может прервать цепочку и сам сформировать ответ клиенту. Один и тот же фильтр может быть применен к нескольким ресурсам. В качестве параметров фильтр получает объекты Request и Response, которые от передает дальше по цепочке или обрывает цепочку и заполняет объект Response.



1. **Слушатели событий (Lister):** серверные объекты – для обработки событий жизненного цикла web-приложения.



1. **Кэш (Cache):** серверный объект, предназначенный для временного хранения данных с целью ускорения выполнения запроса. Кэширование – процессы записи и извлечения данных в/из Cache. Различают кэширование данных и кэширование вывода. **Кэширование данных** – кэширование часто используемых данных. **Кэширование вывода** – кэширование объекта Response.



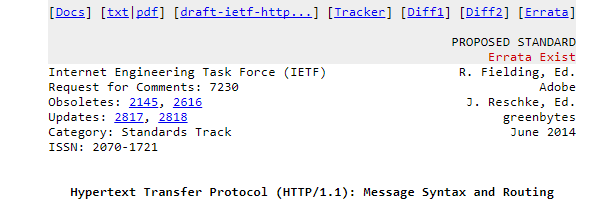
1. **Ресурс-программный код:** экземпляр приложения, который создается для обработки каждого запроса. **HTTP-драйвер** преобразует последовательность битов HTTP-запроса в объект Request. Специальный компонент по запросу (порт, URI,параметры,…)определяет приложение, которое его будет обрабатывать и создает экземпляр приложения.

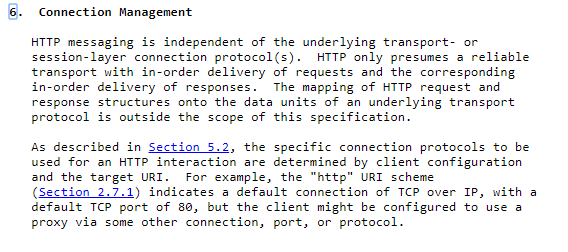


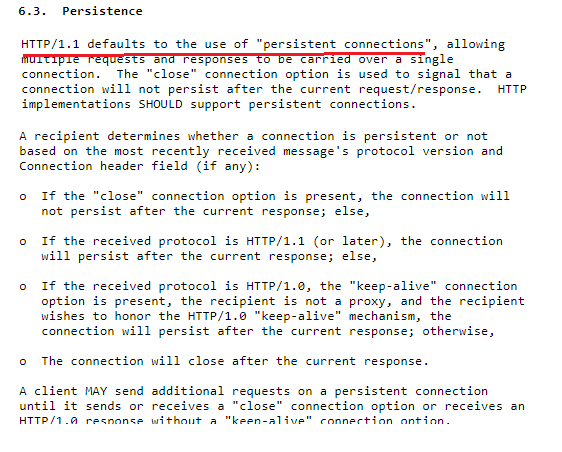


1. **Постоянное соединение:** использование одного TCP-соединения для многократных пар запрос-сервер вместо последовательного открытия новых соединений для каждой пары запрос ответ. Клиент может запросить постоянное соединение с помощью заголовка **Connection: Keep-Alive**, сервер подтверждает заголовком **Connection: Keep-Alive**. Некоторые серверы открывают постоянное соединение по умолчанию и закрывают через заданное время (timeout) бездействия браузера. Большинство браузеров по умолчанию открывают постоянные соединения и отсчитывают timeout для соединения (у разных браузеров разное значение: 60-115с, Opera-системный параметр). Серверы, как правило, поддерживают ограниченное количество постоянных соединений (системный параметр сервера). Обычно по умолчанию используется в HTTPS, так как открытие HTTPS-соединения - очень затратная операция.

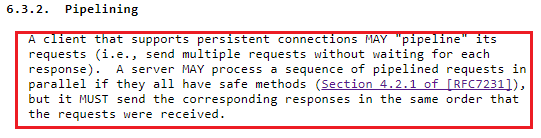


****

****

****

1. **Конвейерная обработка (HTTP pipelining):** технология, которая позволяет в одном соединении передавать несколько различных пар запрос-ответ. Эффективно, если на одной html-странице запрашивается несколько ресурсов с одного сервера. В основном поддерживается браузерами мобильных систем. По протоколу HTTP, серверы должны поддерживать конвейерную обработку запросов, но в реальности этого нет.





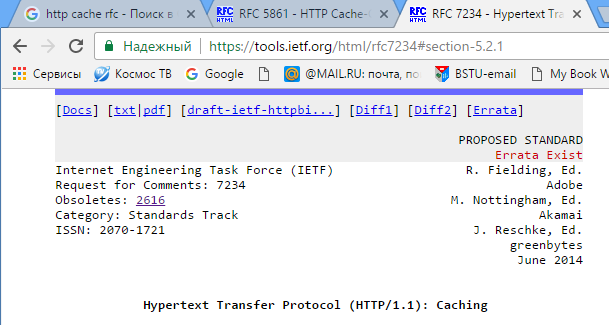


1. **Пул соединений с базой данных:** несколько предварительно и постоянно открытых соединений с сервером СУБД, которые используют приложения. Выбор подключения из пула по Open, возврат в пул Close. Если все подключения пула заняты, запрос на соединение ставится в очередь. Применение пула позволяет увеличить производительность за счет отсутствия процесса подключения к серверу. Альтернатива: держать постоянное открытое соединение для каждого подключения (в общем случае для web-приложения не приемлемо) или открывать/закрывать соединение при каждом запросе (большие накладные расходы на установку соединения). Практически все серверные решения поддерживают автоматические пулы соединений (ASP.NET, спец-драйверы для пула).





1. **Кэширование на стороне браузера:** управление заголовками

****



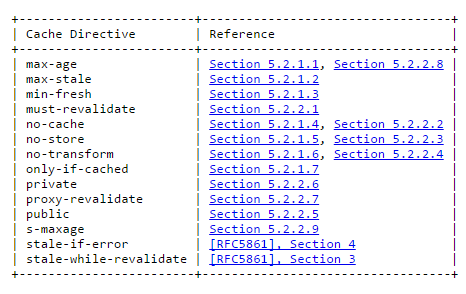




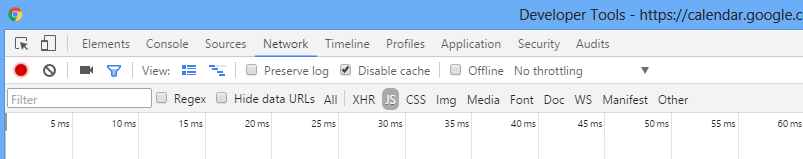




1. **Кэширование на стороне браузера:** HTTPS не кэшируется
2. **Кэширование на стороне браузера: Cache-Control: no-cache** -требует проверку перед каждым использованием кэша.
3. **Кэширование на стороне браузера: Cache-Control: no-store** -требование не использовать кэширование.
4. **Кэширование на стороне браузера (Cache-Control): RFC 7234**

****

1. **Кэширование на стороне браузера:** браузеры позволяют отключить кэширование для отладки.



1. **Синхронный http-запрос:** запрос, при котором поток, выдавший http-запрос, блокируется до поступления запроса.
2. **Асинхронные запросы:** запрос, при котором поток, выдавший http-запрос, не блокируется до поступления запроса; для обработки ответа применяется функция обратного вызова.

1. **AJAX: Asynchronous JavaScript and XML** – асинхронный JavaScript and XML – методология (подход) построения динамических приложений, при которых не осуществляется полная перезагрузка html-страниц. AJAX: XMLHTTPRequest, DOM, формат: XML и JSON.



1. **HTTPS:**
2. **HTTP/2 SPDY**